

IBM RT PC Advanced Interactive Executive Operating System Version 2.1

# AIX Operating System Commands Reference

Programming Family



Personal  
Computer  
Software

SC23-0790-0

# AIX Operating System Commands Reference

**Programming Family**



Personal  
Computer  
Software

---

## **First Edition (January 1987)**

Portions of the code and documentation described in this book were developed at the Electrical Engineering and Computer Sciences Department at the Berkeley Campus of the University of California under the auspices of the Regents of the University of California.

This edition applies to Version 2.1 of IBM RT PC AIX Operating System Licensed Program and to all subsequent releases until otherwise indicated in new editions or technical newsletters. Changes are made periodically to the information herein; these changes will be incorporated in new editions of this publication.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM program product in this publication is not intended to state or imply that only IBM's program product may be used. Any functionally equivalent program may be used instead.

**International Business Machines Corporation provides this manual "as is," without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. IBM may make improvements and/or changes in the products and/or the programs described in this manual at any time.**

Products are not stocked at the address given below. Requests for copies of this product and for technical information about the system should be made to your authorized IBM RT PC dealer.

A reader's comment form is provided at the back of this publication. If the form has been removed, address comments to IBM Corporation, Department 997, 11400 Burnet Road, Austin, Texas 78758-3493. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1985, 1986, 1987

© Copyright INTERACTIVE Systems Corporation 1984

© Copyright AT&T Technologies 1984

---

## About This Book

This book contains reference information on AIX<sup>1</sup> Operating System commands. It describes the commands you can run and summarizes who can call them, how one calls them, what they do, how they read input, how they write output, and how one modifies their actions.

### Who Should Use This Book

To use this book, you should be familiar with AIX or UNIX<sup>2</sup> System 5 commands. If you are not already familiar with AIX or UNIX System V, see *Using the AIX Operating System*. If you are familiar with the commands but need to review how to use the shell and write shell procedures, see “sh” on page 637.

### How To Use This Book

Most of the AIX commands described in this book are in alphabetical order by command name. Some related commands are combined in one description, but these commands should appear in the “Contents” and the “Index.” Therefore, if you cannot find a particular command in this book, the “Contents” or the “Index” should tell you where to find its description.

### Task Index

A “Task Index” on page 7 appears before “Commands” on page 29 to help you locate the commands you need to perform specific tasks. The “Task Index” begins with a “Contents” listing general and specific tasks. To find a command to perform a specific task, go to the page shown and look at the commands and their purposes listed on that page.

---

<sup>1</sup> AIX is a trademark of International Business Machines Corporation.

<sup>2</sup> UNIX was developed and licensed by AT&T. It is a registered trademark of AT&T in the United States of America and other countries.

---

## Commands

“Commands” begin on page 29. The discussion of each command includes the following information:

<b>Purpose</b>	A single-sentence description of the major function of each command.
<b>Syntax</b>	A <i>syntax diagram</i> that shows command line options. For a discussion of how to use this syntax diagram, see “Syntax Diagrams” on page 3.
<b>Description</b>	A discussion of the command that provides more details about its function and use.
<b>Flags</b>	A list of command line flags and associated parameters that explains how they modify the action of the command.
<b>Subcommands</b>	A list of subcommands (for interactive commands) that explains their use.
<b>Examples</b>	Specific examples of how you can use the command.
<b>Files</b>	A list of files used by the command.
<b>Related Information</b>	A list of related commands in this book and related discussions in other books.

For details on other conventions used in this book, see “How To Use The Commands Section” on page 1.

## Special Key Sequences

You can use the AIX Operating System from any of several different display stations, each of which has a different keyboard. In some cases, you must press different keys to perform the same special function from different keyboards. This book identifies both the function name (for example, INTERRUPT) and the necessary key sequence on the IBM RT Personal Computer<sup>3</sup> (in parentheses). If you do not have the IBM RT PC Keyboard, look at your keyboard reference chart to find out which keys on your keyboard produce the special function.

---

<sup>3</sup> RT Personal Computer, RT PC and RT are trademarks of International Business Machines Corporation.

---

## Reference and Information Aids

The standard system devices are described in Appendix A, “AIX Device Table” on page 869. A cross-reference listing of commands and program packages appears in Appendix B, “Program Cross-Reference Index” on page 871. Appendix C, “Details on Reading Syntax Diagrams” on page 879 contains a detailed description of how to read syntax diagrams. A “Glossary” of terms appears after the Appendixes, followed by an “Index.”

A Reader’s Comment Form and Book Evaluation Form are provided at the back of this book. Use the Reader’s Comment Form at any time to give IBM information that may improve the book. After you have become familiar with the book, use the Book Evaluation Form to give IBM specific feedback about the book.

## Prerequisite Information

- *IBM RT PC Using the AIX Operating System* describes using the AIX Operating System commands, working with file systems, and developing shell procedures.
- *IBM RT PC Managing the AIX Operating System* provides instructions for performing such system management tasks as adding and deleting user IDs, creating and mounting file systems, and repairing file system damage.

## Related Information

- *IBM RT PC AIX Operating System Programming Tools and Interfaces* describes the programming environment of the AIX Operating System and includes information about using the operating system tools to develop, compile, and debug programs. In addition, this book describes the operating system services and how to take advantage of them in a program. This book also includes a diskette that includes programming examples, written in C language, to illustrate using system calls and subroutines in short, working programs. (Available optionally)
- *IBM RT PC AIX Operating System Technical Reference* describes the system calls and subroutines that a C programmer uses to write programs for the AIX Operating System. This book also includes information about the AIX file system, special files, file formats, GSL subroutines, and writing device drivers. (Available optionally)
- *IBM RT PC Using AIX Operating System DOS Services* provides step-by-step information for using AIX Operating System DOS Services. (Available optionally; packaged with *IBM RT PC AIX Operating System DOS Services Reference*)
- *IBM RT PC AIX Operating System DOS Services Reference* provides reference information about the AIX Operating System DOS Services. This book also includes information on sharing DOS files with Personal Computer AT Coprocessor Services,

- 
- and on the differences between PC DOS and DOS Services. (Available optionally; packaged with *IBM RT PC Using AIX Operating System DOS Services*)
- *IBM RT PC C Language Guide and Reference* provides guide information for writing, compiling, and running C language programs and includes reference information about C language data structures, operators, expressions, and statements. (Available optionally)
  - *IBM RT PC Messages Reference* lists messages displayed by the IBM RT PC and explains how to respond to the messages.
  - *Virtual Resource Manager Technical Reference* is a two-volume set. The first volume, *Virtual Resource Manager Programming Reference*, describes the VRM programming environment, including the internal VRM routines, VRM floating-point support, use of the VRM debugger, and the supervisor call instructions that form the Virtual Machine Interface. The second volume, *Virtual Resource Manager Device Support*, describes device IPL and configuration, minidisk management, the virtual terminal and block I/O subsystems, as well as the interfaces to the predefined VRM device drivers. This volume also describes the programming conventions for developing your own VRM code and installing it on the system.
  - *IBM RT PC AIX Operating System Text Formatting Guide* describes the functions and capabilities of NROFF and TROFF to perform text processing tasks. (Available optionally)

See *IBM RT PC Bibliography and Master Index* for order numbers of IBM RT PC publications and diskettes.

## Ordering Additional Copies of This Book

To order additional copies of this publication (without program diskettes), use either of the following sources:

- To order from your IBM representative, use Order Number SBOF-0128.
- To order from your IBM dealer, use Part Number 79X3853.

A binder is included with the order. For information on ordering the binder and manual separately, contact your IBM representative or your IBM dealer.

---

# Contents

<b>How To Use The Commands Section</b> .....	<b>1</b>
Command Input and Output .....	2
File Name Substitution .....	2
Syntax Diagrams .....	3
Flag and Parameter Syntax Under Description .....	6
<b>Task Index</b> .....	<b>7</b>
<b>Commands</b> .....	<b>29</b>
<b>acct/*</b> .....	31
<b>chargefee</b> .....	32
<b>ckpacct</b> .....	32
<b>dodisk</b> .....	32
<b>lastlogin</b> .....	33
<b>monacct</b> .....	33
<b>nulladm</b> .....	33
<b>prctmp</b> .....	33
<b>prdaily</b> .....	33
<b>prtacct</b> .....	34
<b>remove</b> .....	34
<b>shutacct</b> .....	34
<b>startup</b> .....	34
<b>turnacct</b> .....	34
<b>acctcms</b> .....	36
<b>acctcom</b> .....	38
<b>acctcon</b> .....	42
<b>acctcon1</b> .....	42
<b>acctcon2</b> .....	43
<b>acctdisk</b> .....	44
<b>acctmerg</b> .....	46
<b>acctprc</b> .....	48
<b>acctprc1</b> .....	48
<b>acctprc2</b> .....	49
<b>accton</b> .....	49
<b>actman</b> .....	50
<b>adb</b> .....	50.1
<b>admin</b> .....	51
<b>ar</b> .....	58
<b>arithmetic</b> .....	62



---

as	64
at, batch	66
awk	70
back	75
backup	76
banner	80
basename, dirname	81
bc	83
bdiff	88
bfs	90
bj	94
bs	95
cal	106
calendar	107
cat	109
cb	111
cc	112
cd	121
cdc	123
cflow	125
chgrp	126.1
chmod	128
chown	132
chperm	133
chroot	134
clri	136
cmp	138
col	140
comb	142
comm	144
confer	146
config	150
connect	152
cp	156
cpio	158
cpp	163
craps	167
crash	168
cron	172
crontab	174
csch	177
csplit	202
ctab	204
ctags	208
cut	210
cvid	212

<b>cw, checkcw</b> .....	213
<b>cxref</b> .....	217
<b>date</b> .....	219
<b>dc</b> .....	222
<b>dcopy</b> .....	226
<b>dd</b> .....	228
<b>defkey</b> .....	232
<b>del</b> .....	234
<b>delta</b> .....	236
<b>deroff</b> .....	239
<b>devices</b> .....	241
<b>devnm</b> .....	242
<b>df</b> .....	244
<b>diff</b> .....	246
<b>diff3</b> .....	249
<b>diffmk</b> .....	252
<b>diremp</b> .....	254
<b>diskusg</b> .....	256
<b>display</b> .....	258
<b>dos</b> .....	262
<b>dosdel</b> .....	266
<b>dosdir</b> .....	267
<b>dosread</b> .....	269
<b>doswrite</b> .....	271
<b>dsipc</b> .....	272.1
<b>dsldxprof</b> .....	272.2
<b>dsstate</b> .....	272.4
<b>dsxlate</b> .....	272.6
<b>du</b> .....	273
<b>dump</b> .....	275
<b>dumpfmt</b> .....	277
<b>echo</b> .....	278
<b>ed</b> .....	280
<b>edit</b> .....	292
<b>env</b> .....	298
<b>eqn, neqn, checkeq</b> .....	300
<b>errdead</b> .....	302
<b>errdemon</b> .....	303
<b>errpt, errpd</b> .....	305
<b>errstop</b> .....	309
<b>errupdate</b> .....	310
<b>ex</b> .....	312
<b>expr</b> .....	317
<b>factor</b> .....	321
<b>ff</b> .....	322
<b>file</b> .....	324

<b>find</b>	326
<b>fish</b>	330
<b>format</b>	331
<b>fortune</b>	332
<b>fptype</b>	332.1
<b>fsck, dfscck</b>	333
<b>fsdb</b>	338
<b>fuser</b>	343
<b>fwtmp</b>	345
<b>acctwtmp</b>	345
<b>wtmpfix</b>	346
<b>gdev</b>	347
<b>hpd</b>	347
<b>erase</b>	348
<b>hardcopy</b>	348
<b>tekset</b>	348
<b>td</b>	348
<b>ged</b>	350
<b>gend</b>	357
<b>get</b>	359
<b>getopt</b>	367
<b>gettext</b>	370
<b>getty</b>	372
<b>graph</b>	375
<b>graphics</b>	377
<b>greek</b>	379
<b>grep</b>	381
<b>groups</b>	385
<b>gutil</b>	386
<b>bel</b>	387
<b>cvrtopt</b>	387
<b>gd</b>	388
<b>gtop</b>	388
<b>pd</b>	388
<b>ptog</b>	388
<b>quit</b>	388
<b>remcom</b>	388
<b>whatis</b>	388
<b>yoo</b>	389
<b>hangman</b>	390
<b>help</b>	391
<b>hp</b>	392
<b>hyphen</b>	394
<b>id</b>	395
<b>init</b>	396
<b>install</b>	399

<b>installp</b> .....	402
<b>inusave</b> .....	403
<b>inurecv</b> .....	404
<b>inurest</b> .....	405
<b>ckprereq</b> .....	406
<b>mvmd</b> .....	407
<b>ipcrm</b> .....	409
<b>ipcs</b> .....	411
<b>ipctable</b> .....	414.1
<b>istat</b> .....	415
<b>join</b> .....	417
<b>keyboard</b> .....	421
<b>kill</b> .....	422
<b>killall</b> .....	425
<b>ld</b> .....	427
<b>lex</b> .....	432
<b>li</b> .....	437
<b>line</b> .....	443
<b>link, unlink</b> .....	444
<b>lint</b> .....	446
<b>ln</b> .....	450
<b>locator</b> .....	452
<b>login</b> .....	453
<b>logname</b> .....	456
<b>lorder</b> .....	457
<b>lp</b> .....	459
<b>ls</b> .....	461
<b>m4</b> .....	465
<b>mail</b> .....	470
<b>make</b> .....	474
<b>makekey</b> .....	481
<b>mdrc</b> .....	482
<b>mesg</b> .....	484
<b>minidisks</b> .....	485
<b>mkdir</b> .....	486
<b>mkfs</b> .....	487
<b>mknod</b> .....	490
<b>mm, checkmm</b> .....	492
<b>mmt, checkmm</b> .....	495
<b>moo</b> .....	497
<b>mount</b> .....	498
<b>mv</b> .....	502
<b>mvdir</b> .....	504
<b>ncheck</b> .....	505
<b>ndtable</b> .....	506.1
<b>newform</b> .....	507

---

<b>newgrp</b>	510
<b>news</b>	512
<b>nice</b>	515
<b>nl</b>	517
<b>nm</b>	521
<b>nohup</b>	523
<b>nroff</b>	525
<b>number</b>	537
<b>od</b>	538
<b>open</b>	541
<b>pack</b>	543
<b>pcat</b>	544
<b>unpack</b>	544
<b>passwd</b>	546
<b>paste</b>	547
<b>penable</b>	550
<b>pdisable</b>	551
<b>phold</b>	551
<b>pg</b>	553
<b>piobe</b>	557
<b>pr</b>	561
<b>profiler</b>	564
<b>prfld</b>	565
<b>prfstat</b>	565
<b>prfdc, prfsnap</b>	565
<b>prfpr</b>	565
<b>print</b>	566
<b>prof</b>	571
<b>proto</b>	573
<b>prs</b>	574
<b>ps</b>	579
<b>ptx</b>	584
<b>puttext</b>	586
<b>pwck</b>	588
<b>pwd</b>	589
<b>qdaemon</b>	590
<b>quiz</b>	591
<b>rc</b>	594
<b>regcmp</b>	595
<b>restore</b>	596
<b>rm</b>	601
<b>rmdel</b>	604
<b>rmdir</b>	605
<b>runacct</b>	606
<b>sact</b>	609
<b>sadc</b>	610

sal	611
sa2	611
sag	612
sar	614
sccsdiff	618
sdb	619
sdiff	627
sed	629
setdma	634.1
setmnt	635
sh	637
shlib	660
shutdown	663
size	665
skulker	667
sleep	668
sno	670
sort	672
sound	679
spell	681
spline	684
split	686
splp	687
stat	690
strip	716
stty	717
su	724
sum	726
sync	727
tab, untab	728
tabs	729
tail	732
tapechk	734
tar	735
tbl	739
tc	742
tctl	744
tee	746
termdef	748
test	750
tic	753
time	754
timex	755
toc	757
dtoc	757
ttoc	758

vtoc	758
touch	760
tplot	762
tput	763
tr	765
trace	768
trcrpt	772
trcstop	774
trcupdate	775
true	777
tsort	778
ttt	780
tty	781
turnon	783
ugtable	784
umask	784.1
umount	786
uname	788
unget	790
uniq	792
units	793
updatep	796
inudocm	799
inuupdt	800
users	802
uuclean	805
uucp	807
uulog	809
uuname	809
uustat	810
uusub	813
uuto	815
uupick	816
uux	818
val	821
varyon	823
vc	826
verify	830
vi, vedit, view	832
vrmdir	842
wait	844
wall	845
wc	846
what	848
who	850
write	853

---

<b>wump</b>	856
<b>xargs</b>	857
<b>yacc</b>	861
<b>300</b>	863
<b>4014</b>	865
<b>450</b>	866
<b>Appendix A. AIX Device Table</b>	<b>869</b>
<b>Appendix B. Program Cross-Reference Index</b>	<b>871</b>
<b>Appendix C. Details on Reading Syntax Diagrams</b>	<b>879</b>
File Input	880
Syntax Diagrams	881
<b>Figures</b>	<b>889</b>
<b>Glossary</b>	<b>891</b>
<b>Index</b>	<b>909</b>





---

# How To Use The Commands Section

This book contains reference information about AIX commands. This information includes syntax diagrams to illustrate how to enter the commands on the command line, descriptions of how commands work, descriptions of command flags and subcommands, lists of related files, and cross-references to related commands in this book and related material in other books.

The following includes a description of:

- Command input and output
- File name substitution by the shell
- Syntax diagrams
- Flag and parameter syntax.

---

## Command Input and Output

Many commands take their input from *standard input* and write their output to *standard output*. By default, standard input comes from the keyboard, and standard output goes to the display. It is important to remember this as you read the command descriptions, since they describe the default action. In this context, the verb *display* means “write to the standard output.” Any command that reads standard input and writes to standard output can have its input or output redirected to a file and can be used in a *pipeline*, where the standard output of a previous command is directed to the standard input of the next command. For more information on pipelines, see “**sh**” on page 637.

There are a few commands that must have a file name supplied or that must read standard input. You can see what a particular command can read by looking at the syntax diagram at the beginning of the description of the command. For instructions on interpreting syntax diagrams, see “Syntax Diagrams” on page 3.

## File Name Substitution

When *file* is supplied as an argument to either a command or a flag, you can automatically produce a list of file name arguments by specifying a pattern that the shell matches against the file names in a directory.

Most characters in such a pattern match themselves, but you can also use some special *pattern-matching characters* in your pattern. These special characters are:

- \* Matches any string, including the null string.
- ? Matches any one character.
- [ . . . ] Matches any one of the characters enclosed in square brackets.
- [! . . . ] Matches any character *other than* one of the characters that follow the exclamation mark within square brackets.

Inside square brackets, a pair of characters separated by a - (minus) specifies a set of all characters lexically within the inclusive range of that pair, so that [a-dy] is equivalent to [abcdy].

Using pattern-matching characters in file names on the command line has some restrictions. If the first character of a file name is a . (dot), it can be matched only by a pattern that begins with a dot. For example, \* matches the file names *myfile* and *yourfile*, but not *.myfile* and *.yourfile*. To match these file names, use a pattern such as:

```
.*file
```

If a pattern does not match any file names, the pattern itself is returned as the result of the match.

---

**Note:** File and directory names should not contain the characters \*, ?, [, or ] because this may create infinite loops during pattern matching attempts.

## Syntax Diagrams

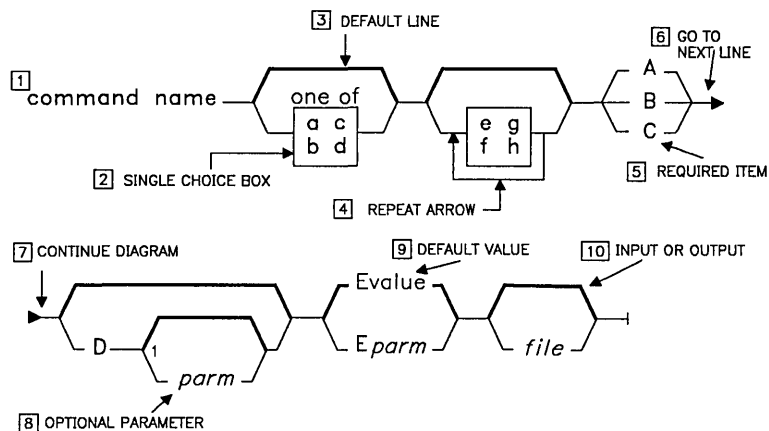
Before each command discussion, you will find a syntax diagram that shows you how to enter that command correctly on the command line. These diagrams show:

- Which flags can be entered on the command line
- Which flags must take parameters
- Which flags have optional parameters
- Default values of flags and parameters, if any
- Which flags can and cannot be entered together
- Where you must enter flags or parameters and where you have a choice
- Where you can repeat flag and parameter sequences.

The following discussion explains how to interpret the syntax diagrams. It begins with an example diagram that shows most of the conventions used in diagrams. Each part of the diagram is labeled and explained. Following the example are sample diagrams taken from this book.

Diagram items that must be entered literally on the command line are in **bold**. These items include the command name, all flags, and literal characters. Variable items are in *italics*. These items include parameters that follow flags, and parameters that the command reads, such as *files* and *directories*. If an item has a default value, it is shown in the normal font and the path is shown in bold. You do not enter on the command line any item shown in the normal font on a bold path.

The following diagram is an example to illustrate the conventions used in the syntax diagrams:



OL805370

<sup>1</sup> Do not put a blank between these items.

OL805308

You interpret the diagram as follows:

- |                       |   |
|-----------------------|---|
| 1 <b>command name</b> | The first item in the diagram is the name of the command you want to invoke. It is in bold, so it must be entered exactly as it appears in the diagram.   |
| 2 SINGLE CHOICE BOX   | If you follow the lower path, you encounter a box with the words one of over it. You can choose only one item from this box.  |
| 3 DEFAULT LINE        | If you follow the upper path, you bypass the single choice box, and enter nothing. The bold line around the box is a default line, which means that you do not have to enter anything from that part of the diagram. Exceptions are usually explained under "Description." One important exception, the blank default line around input and output files, is explained in item 10.                                |
| 4 REPEAT ARROW        | When you follow a path that takes you to a box with an arrow around it, you must choose at least one item from the box. Then you can either follow the arrow back around and continue to choose items from it, or you can continue along the path. When following the arrow around just the box (rather than an arrow that includes several branches in the diagram), do not choose the same item more than once. |

- 
- 5 REQUIRED ITEM Following the branch with the repeat arrow is a branch with three choices and no default line around them. This means that you must choose one of **A**, **B**, or **C**.
- 6 GO TO NEXT LINE If a diagram is too long to fit on one line, this character tells you to go to the next line of the diagram to continue entering your command line. Remember, the diagram does not end until you reach the vertical mark.
- 7 CONTINUE DIAGRAM This character shows you where to continue with the diagram after it breaks on the previous line.
- 8 OPTIONAL PARAMETER If a flag can, but does not have to, take a parameter, the path branches after the flag to show this. If you cannot enter a space between the flag and parameter, you are told in a footnote.
- 9 DEFAULT VALUE Often, a command has default values or actions that it will follow if you do not enter a specific item. These default values are indicated in normal font in the default line if they are equivalent to something you could enter on the command line (for example, a flag with a value). If the default is not something you can enter on the command line, it is not indicated in the diagram. However, it is discussed under “Flags.”
- Note:** Default values are included in the diagram for your information. Do not enter them on the command line.
- 10 INPUT OR OUTPUT A command that can read either standard input or input files has an empty default line around the file parameter. If the command can write its output to either a file or to standard output, it is also shown with a default line around the output file parameter. If a command can read only from standard input, input is not shown in the diagram, and standard input is assumed. If a command writes only to standard output, this is also assumed and output is not included in the diagram. When you must supply a file name for input or output, the file parameter is included in the diagram without a default line around it.

Following are examples of how to enter this command based on this syntax diagram.

```
command name A
command name C
command name a B
command name d B
command name e A
command name e g f A
command name C D
command name C D8
command name A E7
command name B myfile
command name a e g B D3 E6 myfile
command name d f e h C D myfile
```

---

**Note:** Although the diagram implies that the order of the flags is important, it is usually not. When the order of the flags is important, it is indicated in the diagram, under “Flags,” or in both places. With this in mind, an additional example of how to enter this command is:

```
command name E9 a D g A h f myfile
```

This is a brief discussion of how to read syntax diagrams. If you need more information on the syntax diagrams, Appendix C, “Details on Reading Syntax Diagrams” on page 879 shows several more examples from this book, and explains in detail how to interpret the diagrams.

## Flag and Parameter Syntax Under Description

The description of flags and parameters under “Description” uses the following conventions:

- bold**        Flags and other items in bold are to be entered literally.
- italics*     Items in italics are parameter names for which you substitute an appropriate value in that position on the command line. For example, if you see *file*, you should type in the name of a file in that position.
- [ ]            Items in brackets are optional. The only exception is brackets that are themselves in bold. Brackets in bold are part of what is to be entered literally.
- . . .          Items followed by an ellipsis can be repeated. Thus, if you see *file . . .*, you can type several file names separated by blanks.

Using these conventions, the following string:

```
-Dname[=value]
```

shows that, with the **-D** flag, the *name* parameter is required but assigning a *value* to *name* is optional. The following are valid ways to specify this flag and parameter combination:

```
-Daxis  
-Daxis=10
```

The next string shows a parameter that can be replaced by several values:

```
-l file . . .
```

The following are valid ways to enter the **-l** flag:

```
-l memo letter  
-l memo  
-l letter
```

---

# Task Index

This index lists most of the commands that are described in this book and gives the purpose of each command. The commands are grouped by task to help you find a command by the task it performs.

Operating and Managing the System .....	9
Starting and Stopping the System .....	9
Installing Programs .....	9
Configuring the System .....	9
Controlling System Processes .....	10
Displaying and Printing Data and Text .....	11
Performing Calculator Functions .....	11
Controlling System Security .....	12
Displaying System Statistics and Information .....	12
Performing System Accounting Functions .....	14
Managing File Systems .....	14
Analyzing System Activity .....	15
Backing Up and Restoring System Files .....	16
Working with Files and Directories .....	16
Working with Directories .....	16
Archiving Files .....	16
Comparing Files .....	17
Copying and Moving Files .....	17
Creating and Editing Files .....	17
Deleting Files .....	18
Merging and Splitting Files .....	18
Printing and Displaying Files .....	18
Scanning Files .....	19
Sorting Files .....	19
Displaying, Setting, and Changing File Permission .....	19
Working with Data .....	20
Working with Text .....	20
Creating and Editing Text Files .....	20
Formatting Text .....	21
Communicating with Other Users .....	22
Communicating with Other Systems .....	22
Working with Disks and Diskettes .....	22
Working with Tape .....	23
Working with Work Stations .....	23
Working with Graphics .....	24



---

Developing Programs .....	24
Programming in Assembler .....	24
Programming in C .....	24
Programming in Miscellaneous Languages .....	25
Programming in Shell .....	25
Handling Messages .....	26
Debugging Programs .....	26
Managing Programs .....	26
Making and Installing Programs .....	26
Managing Source Programs Using the Source Code Control System (SCCS) .....	26
Managing Object Files .....	27
Playing Games .....	28

---

# Operating and Managing the System

## Starting and Stopping the System

<b>actman</b>	Lets you interact with multiple virtual terminals.
<b>date</b>	Displays or sets the date.
<b>getty</b>	Sets the characteristics of ports.
<b>init</b>	Initializes the system.
<b>login</b>	Allows you to sign on to the system.
<b>newgrp</b>	Changes your primary group identification.
<b>open</b>	Opens a virtual terminal.
<b>passwd</b>	Changes your login password.
<b>penable</b>	Controls or reports the availability of login ports.
<b>rc</b>	Performs normal startup initialization.
<b>shutdown</b>	Ends system operation.
<b>users</b>	Adds, deletes and changes user and group information.

## Installing Programs

<b>ckprereq</b>	Checks the level of the prerequisite licensed program.
<b>cvid</b>	Creates a VRM install diskette for backup purposes.
<b>install</b>	Installs a command.
<b>installp</b>	Installs a program.
<b>mvmd</b>	Updates the VRM minidisk.
<b>updatep</b>	Updates one or more programs.
<b>vrconfig</b>	Installs peripheral devices.

## Configuring the System

<b>chparm</b>	Changes or examines system parameters.
<b>config</b>	Extracts configuration information from configuration files.
<b>devices</b>	Adds, deletes, changes, and displays device information.
<b>defkey</b>	Defines keyboard key assignments.

<b>display</b>	Selects the physical display that an existing or new virtual terminal uses and sets colors and fonts.
<b>dsipc</b>	Installs the Interprocess Communication key mapping in the kernel.
<b>dsxlate</b>	Installs Distributed Services UID/GID translate tables into the kernel.
<b>dslxprof</b>	Loads translate information into the UID/GID translate profiles.
<b>env</b>	Sets the environment for execution of a command.
<b>getty</b>	Sets the characteristics of ports.
<b>keyboard</b>	Controls the delay and repetition rates of the keyboard.
<b>locator</b>	Controls the sample rate of the locator.
<b>minidisks</b>	Adds, deletes, changes, and displays minidisks.
<b>mdrc</b>	Allows you to reinstall a user-created minidisk after you have reinstalled AIX.
<b>mknod</b>	Creates a special file.
<b>mvmd</b>	Updates the VRM minidisk.
<b>penable</b>	Controls or reports the availability of login ports.
<b>pdisable</b>	Kills the logger running on the specified port.
<b>phold</b>	Prevents new users from logging into a port.
<b>sound</b>	Controls the volume and click of the keyboard speaker.
<b>splp</b>	Changes or displays printer driver settings.
<b>stty</b>	Sets, resets, or reports work station operating parameters.
<b>termdef</b>	Queries terminal characteristics.
<b>users</b>	Adds, deletes and changes user and group information.
<b>varyon</b>	Makes an IBM 9332 Direct Access Storage Device and the minidisks and file systems defined on it available for use.
<b>verify</b>	Turns write verification on or off for a particular minidisk.
<b>vrconfig</b>	Installs peripheral devices.

## Controlling System Processes

<b>at</b>	Runs commands at a later time.
<b>cron</b>	Runs commands automatically.
<b>crontab</b>	Submits a schedule of commands to <b>cron</b> .
<b>errdemon</b>	Starts the error-logging demon.
<b>errstop</b>	Terminates the error-logging demon.

---

<b>kill</b>	Sends a signal to a running process.
<b>killall</b>	Cancels all processes except the calling process.
<b>nice</b>	Runs a command at a different priority.
<b>nohup</b>	Runs a command without hangups and quits.
<b>open</b>	Opens a virtual terminal.
<b>pr</b>	Displays a Source Code Control System (SCCS) file.
<b>qdaemon</b>	Schedules jobs enqueued by the <b>print</b> command.
<b>sleep</b>	Suspends execution for an interval.
<b>trace</b>	Starts the trace function.
<b>trcstop</b>	Stops the trace function.
<b>wait</b>	Waits for completion of a process.

### Displaying and Printing Data and Text

<b>banner</b>	Writes character strings in large letters to standard output.
<b>cal</b>	Displays a calendar.
<b>cat</b>	Concatenates or displays files.
<b>calendar</b>	Writes reminder messages to standard output.
<b>col</b>	Processes text having reverse linefeeds and forward/reverse half-linefeeds for output to standard output.
<b>echo</b>	Writes its arguments to standard output.
<b>lp</b>	Prints a file in a format suitable for sending to a line printer.
<b>pg</b>	Formats files to the work station.
<b>pr</b>	Writes a file to standard output.
<b>piobe</b>	Writes a file to standard output in a format suitable for sending to a line printer.
<b>print</b>	Enqueues a file.
<b>qdaemon</b>	Schedules jobs enqueued by the <b>print</b> command.
<b>splp</b>	Changes or displays printer driver settings.

### Performing Calculator Functions

<b>bc</b>	Provides an interpreter for arbitrary-precision arithmetic language.
<b>dc</b>	Provides an interactive desk calculator for doing arbitrary-precision integer arithmetic.

**factor** Factors a number.

## **Controlling System Security**

**chgrp** Changes the group ownership of a file or directory.

**chmod** Changes permission codes.

**chown** Changes the owner of files or directories.

**id** Displays the system identity of the user issuing the command.

**logname** Displays your login name.

**login** Allows you to sign on to the system.

**newgrp** Changes your primary group identification.

**passwd** Changes your login password.

**pwck** Checks the password and group files for inconsistencies.

**su** Obtains the privileges of another user, including superuser authority.

**umask** Sets file-creation permission code mask.

**users** Adds, deletes and changes user and group information.

## **Displaying System Statistics and Information**

**chparm** Changes or examines system parameters.

**date** Displays or sets the date.

**devices** Adds, deletes, changes, and displays device information.

**diskusg** Generates disk accounting data by user ID.

**dsstate** Sets the state of the Distributed Services kernel logic.

**errpt** Processes a report of logged errors.

**errupdate** Updates an error report template.

**file** Determines file type.

**fuser** Identifies processes using a file or file structure.

**groups** Displays your group membership.

**help** Provides information about a Source Code Control System (SCCS) message or command or about certain non-SCCS commands.

**id** Displays the system identity of the user issuing the command.

**ipcs** Reports inter-process communication facility status.

**istat** Examines i-nodes.

**logname** Displays your login name.

---

<b>minidisks</b>	Adds, deletes, changes, and displays minidisks.
<b>ncheck</b>	Generates path names from i-numbers.
<b>news</b>	Writes system news items to standard output.
<b>od</b>	Writes the contents of storage to the standard output.
<b>penable</b>	Controls or reports the availability of login ports.
<b>prfld</b>	Profiles the operating system.
<b>profiler</b>	Displays program profile data.
<b>ps</b>	Reports process status.
<b>pwck</b>	Checks the password and group files for inconsistencies.
<b>pwd</b>	Displays the path name of the working directory.
<b>sact</b>	Displays current Source Code Control System (SCCS) file editing status
<b>sadc</b>	Provides a system activity report package.
<b>sag</b>	Displays a graph of system activity.
<b>sar</b>	Collects, reports, or saves system activity information.
<b>splp</b>	Changes or displays printer driver settings.
<b>stty</b>	Sets, resets, or reports work station operating parameters.
<b>sum</b>	Displays the checksum and block count of a file.
<b>time</b>	Times the execution of a command.
<b>timex</b>	Times a command, and reports process data and system activity.
<b>tty</b>	Writes to standard output the full path name of your work station.
<b>uname</b>	Displays the name of the current operating system.
<b>uustat</b>	Reports the status of and provides rudimentary job control for the <b>uucp</b> command.

---

**uusub** Defines and monitors a **uucp** subnetwork structure.  
**who** Identifies the users currently logged in.

## Performing System Accounting Functions

**acctdisk** Performs disk-usage accounting.  
**acctcms** Produces command usage summaries from accounting records.  
**acctcom** Displays selected process accounting record summaries.  
**acctcon1** Performs connect-time accounting.  
**acctmerg** Merges total accounting files.  
**acctprcl** Performs process accounting.  
**acct/\*** Provides accounting shell procedures.  
**diskusg** Generates disk accounting data by user ID.  
**du** Summarizes disk usage.  
**fwtmp** Manipulates connect accounting records.  
**runacct** Runs daily accounting.  
**sadc** Provides a system activity report package.

## Managing File Systems

**basename** Returns the base name of a string parameter.  
**chroot** Changes the root directory of a command.  
**clri** Clears the specified i-node.  
**cpio** Copies files into and out of archive storage and directories.  
**crash** Examines system images.  
**dcopy** Copies file systems for the best access time.  
**devnm** Names a device.  
**df** Reports number of available disk blocks.  
**du** Summarizes disk usage.  
**ff** Lists the file names and statistics for a file system.  
**fsck** Checks file system consistency and interactively repairs the file system.  
**fsdb** Debugs file systems.  
**fuser** Identifies processes using a file or file structure.

---

<b>env</b>	Sets the environment for execution of a command.
<b>istat</b>	Examines i-nodes.
<b>link</b>	Performs the link system call.
<b>mdrc</b>	Allows you to reinstall a user-created minidisk after you have reinstalled AIX.
<b>mkfs</b>	Makes a file system.
<b>mknod</b>	Creates a special file.
<b>mount</b>	Makes a file system available for use.
<b>ncheck</b>	Generates path names from i-numbers.
<b>proto</b>	Constructs a prototype file for a file system.
<b>setmnt</b>	Creates mount table.
<b>skulker</b>	Cleans up file systems by removing unwanted files.
<b>sync</b>	Updates the superblock and writes buffered files to the fixed disk.
<b>unlink</b>	Performs the unlink system call.
<b>umount</b>	Makes a file system unavailable for use.
<b>varyon</b>	Makes an IBM 9332 Direct Access Storage Device and the minidisks and file systems defined on it available for use.

### Analyzing System Activity

<b>errdead</b>	Extracts error records from dump.
<b>errdemon</b>	Starts the error-logging demon.
<b>errpt</b>	Processes a report of logged errors.
<b>errstop</b>	Terminates the error-logging demon.
<b>errupdate</b>	Updates an error report template.
<b>dumpfmt</b>	Formats the VRM dump file.
<b>fuser</b>	Identifies processes using a file or file structure.
<b>prfld</b>	Profiles the operating system.
<b>time</b>	Times the execution of a command.
<b>trace</b>	Starts the trace function.
<b>trcrpt</b>	Formats a report from the trace log file.
<b>trcstop</b>	Stops the trace function.
<b>trcupdate</b>	Updates trace format templates.



---

## Backing Up and Restoring System Files

<b>backup</b>	Backs up files.
<b>pack</b>	Compresses files.
<b>restore</b>	Copies back files created by the <b>backup</b> command.
<b>tapechk</b>	Performs consistency checking of the streaming tape device.
<b>tar</b>	Manipulates tape archives.
<b>tctl</b>	Gives commands to streaming tape.

## Working with Files and Directories

### Working with Directories

<b>cd</b>	Changes the current directory.
<b>chroot</b>	Changes the root directory of a command.
<b>dircmp</b>	Compares two directories and the contents of their common files.
<b>dosdir</b>	Lists the directory for DOS files.
<b>find</b>	Finds files matching expression.
<b>li</b>	Lists the contents of a directory.
<b>ls</b>	Displays the contents of a directory.
<b>mkdir</b>	Makes a directory.
<b>mvdir</b>	Moves (renames) a directory.
<b>pwd</b>	Displays the path name of the working directory.
<b>rm</b>	Removes files or directories.
<b>rmdir</b>	Removes a directory.

### Archiving Files

<b>ar</b>	Maintains portable libraries used by the linkage editor.
<b>backup</b>	Backs up files.
<b>cpio</b>	Copies files into and out of archive storage and directories.
<b>lorder</b>	Finds the best order for member files in an object library.
<b>pack</b>	Compresses files.

---

<b>restore</b>	Copies back files created by the <b>backup</b> command.
<b>shlib</b>	Creates a shared library.
<b>tar</b>	Manipulates tape archives.

## Comparing Files

<b>cmp</b>	Compares two files.
<b>comm</b>	Selects or rejects lines common to two sorted files.
<b>bdiff</b>	Uses <b>diff</b> to find differences in very large files.
<b>diff</b>	Compares text files.
<b>diff3</b>	Compares three files.
<b>diffmk</b>	Marks differences between files.
<b>dircmp</b>	Compares two directories and the contents of their common files.
<b>sdiff</b>	Compares two files and displays the differences in a side-by-side format.
<b>scsdiff</b>	Compares two versions of a Source Code Control System (SCCS) file.
<b>uniq</b>	Deletes repeated lines in a file.

## Copying and Moving Files

<b>cat</b>	Concatenates or displays files.
<b>cp</b>	Copies files.
<b>dd</b>	Converts and copies a file.
<b>dosread</b>	Copies a DOS file.
<b>doswrite</b>	Copies AIX files to DOS files.
<b>ln</b>	Links files.
<b>mv</b>	Moves files.
<b>uucp</b>	Copies files from one AIX system to another.
<b>uuto</b>	Copies public files from one AIX system to another system with local system control of file access.

## Creating and Editing Files

<b>admin</b>	Creates and initializes SCCS files.
<b>cdc</b>	Changes the comments in a Source Code Control System (SCCS) delta.
<b>ed</b>	Edits text by line.

---

<b>edit</b>	Provides a simple line editor for the new user.
<b>ex</b>	Edits lines interactively, with screen display.
<b>get</b>	Creates a specified version of a Source Code Control System (SCCS) file.
<b>mknod</b>	Creates a special file.
<b>sed</b>	Provides a stream editor.
<b>uniq</b>	Deletes repeated lines in a file.
<b>vi</b>	Edits files with a full screen display.

### Deleting Files

<b>del</b>	Deletes files if the request is confirmed.
<b>dosdel</b>	Deletes DOS files.
<b>rm</b>	Removes files or directories.
<b>uniq</b>	Deletes repeated lines in a file.
<b>uuclean</b>	Deletes from the <b>uucp</b> spool directory or a named directory selected files older than a specified number of hours.

### Merging and Splitting Files

<b>csplit</b>	Splits files by context.
<b>cut</b>	Writes out selected fields from each line of a file.
<b>join</b>	Joins data fields of two files.
<b>paste</b>	Merges the lines of several files or subsequent lines in one file.
<b>sort</b>	Sorts or merges files.
<b>split</b>	Splits a file into pieces.
<b>tail</b>	Writes a file to standard output, beginning at a specified point.

### Printing and Displaying Files

<b>cat</b>	Concatenates or displays files.
<b>cut</b>	Writes out selected fields from each line of a file.
<b>lp</b>	Prints a file in a format suitable for sending to a line printer.
<b>nl</b>	Numbers lines in a file.
<b>od</b>	Writes the contents of storage to the standard output.
<b>pg</b>	Formats files to the work station.

---

<b>piobe</b>	Writes a file to standard output in a format suitable for sending to a line printer.
<b>pr</b>	Writes a file to standard output.
<b>prs</b>	Displays a Source Code Control System (SCCS) file.
<b>print</b>	Enqueues a file.
<b>qdaemon</b>	Schedules jobs enqueued by the <b>print</b> command.
<b>splp</b>	Changes or displays printer driver settings.
<b>tail</b>	Writes a file to standard output, beginning at a specified point.

### Scanning Files

<b>awk</b>	Finds lines in files matching specified patterns and performs specified actions on them.
<b>bfs</b>	Scans files.
<b>file</b>	Determines file type.
<b>find</b>	Finds files matching expression.
<b>grep</b>	Searches a file for a pattern.
<b>sed</b>	Provides a stream editor.
<b>spell</b>	Finds spelling errors.
<b>uniq</b>	Deletes repeated lines in a file.
<b>wc</b>	Counts the number of lines, words, and characters in a file.
<b>what</b>	Displays identifying information in files.

### Sorting Files

<b>lorder</b>	Finds the best order for member files in an object library.
<b>sort</b>	Sorts or merges files.
<b>tsort</b>	Sorts an unordered list of ordered pairs (a topological sort).

### Displaying, Setting, and Changing File Permission

<b>chgrp</b>	Changes the group ownership of a file or directory.
<b>chmod</b>	Changes permission codes.
<b>chown</b>	Changes the owner of files or directories.
<b>groups</b>	Displays your group membership.

---

<b>li</b>	Lists the contents of a directory.
<b>ls</b>	Displays the contents of a directory.
<b>umask</b>	Sets file-creation permission code mask.

## Working with Data

<b>cal</b>	Displays a calendar.
<b>calendar</b>	Writes reminder messages to standard output.
<b>echo</b>	Writes its arguments to standard output.
<b>ed</b>	Edits text by line.
<b>edit</b>	Provides a simple line editor for the new user.
<b>ex</b>	Edits lines interactively, with screen display.
<b>join</b>	Joins data fields of two files.
<b>tr</b>	Translates characters.
<b>units</b>	Converts units in one measure to equivalent units in another measure.
<b>vi</b>	Edits files with a full screen display.
<b>wc</b>	Counts the number of lines, words, and characters in a file.

## Working with Text

### Creating and Editing Text Files

<b>ed</b>	Edits text by line.
<b>edit</b>	Provides a simple line editor for the new user.
<b>ex</b>	Edits lines interactively, with screen display.
<b>hyphen</b>	Finds hyphenated words.
<b>spell</b>	Finds spelling errors.
<b>sed</b>	Provides a stream editor.
<b>tab</b>	Changes space characters into tabs.
<b>tr</b>	Translates characters.
<b>untab</b>	Changes tabs into space characters.
<b>vi</b>	Edits files with a full screen display.

---

## Formatting Text

<b>col</b>	Processes text having reverse linefeeds and forward/reverse half-linefeeds for output to standard output.
<b>greek</b>	Converts output for a TELETYPE Model 37 work station to output for other work stations.
<b>hp</b>	Handles special functions for the HP2640- and HP2621-series terminals.
<b>hyphen</b>	Finds hyphenated words.
<b>cw</b>	Prepares constant-width text for <b>troff</b> .
<b>deroff</b>	Removes <b>nroff</b> , <b>troff</b> , <b>tbl</b> , and <b>eqn</b> constructs from files.
<b>diffmk</b>	Marks differences between files.
<b>eqn</b>	Formats mathematical text for the <b>nroff</b> and <b>troff</b> commands.
<b>mm</b>	Displays or checks documents formatted with Memorandum Macros.
<b>mmt</b>	Typesets documents, manual pages, view graphs, and slides.
<b>newform</b>	Changes the format of a text file.
<b>nl</b>	Numbers lines in a file.
<b>nroff</b>	Formats text for printing devices.
<b>paste</b>	Merges the lines of several files or subsequent lines in one file.
<b>ptx</b>	Generates a permuted index.
<b>tab</b>	Changes space characters into tabs.
<b>tabs</b>	Sets tab stops on work stations.
<b>tbl</b>	Formats tables for the <b>nroff</b> and <b>troff</b> commands.
<b>tc</b>	Simulates phototypesetter output for a Tektronix 4014 work station.
<b>troff</b>	Formats text for a phototypesetter.
<b>untab</b>	Changes tabs into space characters.
<b>300</b>	Handles special line-motion functions for DASI 300/300s work stations.
<b>4014</b>	Formats a full page 66-line screen display for a Tektronix 4014 work station.
<b>450</b>	Handles special line-motion functions for the DASI 450 work station.

---

## Communicating with Other Users

<b>confer</b>	Provides an on-line conferencing system.
<b>mail</b>	Sends messages to system users and displays messages from system users.
<b>mesg</b>	Permits or refuses <b>write</b> messages.
<b>news</b>	Writes system news items to standard output.
<b>sum</b>	Displays the checksum and block count of a file.
<b>wall</b>	Writes a message to all logged-in users.
<b>who</b>	Identifies the users currently logged in.
<b>write</b>	Sends messages to other users on the system.

## Communicating with Other Systems

<b>connect</b>	Establishes a connection to a remote system.
<b>sum</b>	Displays the checksum and block count of a file.
<b>uuclean</b>	Deletes from the <b>uucp</b> spool directory or a named directory selected files older than a specified number of hours.
<b>uucp</b>	Copies files from one AIX system to another.
<b>uustat</b>	Reports the status of and provides rudimentary job control for the <b>uucp</b> command.
<b>uusub</b>	Defines and monitors a <b>uucp</b> subnetwork structure.
<b>uuto</b>	Copies public files from one AIX system to another system with local system control of file access.
<b>uux</b>	Runs a command on another AIX system.

## Working with Disks and Diskettes

<b>format</b>	Formats diskettes.
<b>mdrc</b>	Allows you to reinstall a user-created minidisk after you have reinstalled AIX.
<b>minidisks</b>	Adds, deletes, changes, and displays minidisks.
<b>mount</b>	Makes a file system available for use.
<b>umount</b>	Makes a file system unavailable for use.

---

<b>varyon</b>	Makes an IBM 9332 Direct Access Storage Device and the minidisks and file systems defined on it available for use.
<b>verify</b>	Turns write verification on or off for a particular minidisk.

## Working with Tape

<b>tapechk</b>	Performs consistency checking of the streaming tape device.
<b>tar</b>	Manipulates tape archives.
<b>tctl</b>	Gives commands to streaming tape.

## Working with Work Stations

<b>display</b>	Selects the physical display that an existing or new virtual terminal uses and sets colors and fonts.
<b>echo</b>	Writes its arguments to standard output.
<b>hp</b>	Handles special functions for the HP2640- and HP2621-series terminals.
<b>keyboard</b>	Controls the delay and repetition rates of the keyboard.
<b>defkey</b>	Defines keyboard key assignments.
<b>locator</b>	Controls the sample rate of the locator.
<b>penable</b>	Controls or reports the availability of login ports.
<b>stty</b>	Sets, resets, or reports work station operating parameters.
<b>tabs</b>	Sets tab stops on work stations.
<b>termdef</b>	Queries terminal characteristics.
<b>tic</b>	Translates <b>terminfo</b> files from source to compiled format.
<b>tput</b>	Queries the <b>terminfo</b> file.
<b>tty</b>	Writes to standard output the full path name of your work station.
<b>300</b>	Handles special line-motion functions for DASI 300/300s work stations.
<b>4014</b>	Formats a full page 66-line screen display for a Tektronix 4014 work station.
<b>450</b>	Handles special line-motion functions for the DASI 450 work station.



---

## Working with Graphics

<b>gdev</b>	Provides graphical device routines and filters.
<b>ged</b>	Displays, makes, and edits graphical files on Tektronix 4010 terminals.
<b>gend</b>	Provides a general graphics device backend.
<b>graph</b>	Draws a graph.
<b>graphics</b>	Accesses graphical and numerical commands.
<b>gutil</b>	Provides graphical utility programs.
<b>spline</b>	Interpolates smooth curve.
<b>stat</b>	Provides tools for analyzing numerical data.
<b>toc</b>	Provides graphical table of contents routines.
<b>tpplot</b>	Produces plotting instructions for a particular work station.

## Developing Programs

### Programming in Assembler

<b>as</b>	Assembles a source file.
<b>sdb</b>	Provides a symbolic debugger for C and assembler programs.

### Programming in C

<b>cb</b>	Puts C source code into a form that is easily read.
<b>cc</b>	Compiles C programs.
<b>cflow</b>	Generates a C flow graph of external references.
<b>cpp</b>	Performs file inclusion and macro substitution on C Language source files.
<b>cxref</b>	Creates a C program cross-reference listing.
<b>factor</b>	Factors a number.
<b>ipcrm</b>	Removes message queue, semaphore set or shared memory identifiers.
<b>m4</b>	Preprocesses files, expanding macro definitions.
<b>lex</b>	Generates a C Language program that matches patterns for simple lexical analysis of an input stream.
<b>lint</b>	Checks C programs for potential problems.

---

<b>regcmp</b>	Compiles patterns.
<b>sdb</b>	Provides a symbolic debugger for C and assembler programs.
<b>tic</b>	Translates <b>terminfo</b> files from source to compiled format.
<b>yacc</b>	Generates a LR(1) parsing program from input consisting of a context-free grammar specification.

## Programming in Miscellaneous Languages

<b>bc</b>	Provides an interpreter for arbitrary-precision arithmetic language.
<b>bs</b>	Compiles and interprets modest-sized programs.
<b>lex</b>	Generates a C Language program that matches patterns for simple lexical analysis of an input stream.
<b>m4</b>	Preprocesses files, expanding macro definitions.
<b>sno</b>	Provides a SNOBOL interpreter.

## Programming in Shell

<b>basename</b>	Returns the base name of a string parameter.
<b>csch</b>	Interprets commands read from a file or entered from the keyboard.
<b>cron</b>	Runs commands automatically.
<b>crontab</b>	Submits a schedule of commands to <b>cron</b> .
<b>echo</b>	Writes its arguments to standard output.
<b>env</b>	Sets the environment for execution of a command.
<b>expr</b>	Evaluates arguments as an expression.
<b>find</b>	Finds files matching expression.
<b>getopt</b>	Parses command line flags and parameters.
<b>line</b>	Reads one line from the standard input.
<b>nice</b>	Runs a command at a different priority.
<b>nohup</b>	Runs a command without hangups and quits.
<b>open</b>	Opens a virtual terminal.
<b>sh</b>	Interprets commands read from a file or entered at the keyboard.
<b>sleep</b>	Suspends execution for an interval.
<b>tee</b>	Displays the output of a program and copies it into a file.
<b>test</b>	Evaluates conditional expressions.

<b>time</b>	Times the execution of a command.
<b>true</b>	Returns an exit value of zero.
<b>wait</b>	Waits for completion of a process.
<b>xargs</b>	Constructs argument lists and runs commands.

## Handling Messages

<b>gettext</b>	Extracts message/insert/help descriptions.
<b>puttext</b>	Updates an output file that contains message/insert/help descriptions.

## Debugging Programs

<b>crash</b>	Examines system images.
<b>dump</b>	Dumps selected parts of an object file.
<b>dumpfmt</b>	Formats the VRM dump file.
<b>od</b>	Writes the contents of storage to the standard output.
<b>prof</b>	Displays program profile data.
<b>adb</b>	Provides a general purpose debugger.
<b>sdb</b>	Provides a symbolic debugger for C and assembler programs.
<b>time</b>	Times the execution of a command.
<b>timex</b>	Times a command, and reports process data and system activity.

## Managing Programs

### Making and Installing Programs

<b>make</b>	Maintains up-to-date versions of programs.
<b>install</b>	Installs a command.
<b>installp</b>	Installs a program.
<b>updatep</b>	Updates one or more programs.

### Managing Source Programs Using the Source Code Control System (SCCS)

<b>admin</b>	Creates and initializes SCCS files.
<b>cdc</b>	Changes the comments in a Source Code Control System (SCCS) delta.
<b>comb</b>	Combines SCCS deltas.

---

<b>delta</b>	Creates a delta in a Source Code Control System file.
<b>get</b>	Creates a specified version of a Source Code Control System (SCCS) file.
<b>help</b>	Provides information about a Source Code Control System (SCCS) message or command or about certain non-SCCS commands.
<b>prs</b>	Displays a Source Code Control System (SCCS) file.
<b>rmdel</b>	Removes a delta from a Source Code Control System (SCCS) file.
<b>sact</b>	Displays current Source Code Control System (SCCS) file editing status.
<b>sccsdiff</b>	Compares two versions of a Source Code Control System (SCCS) file.
<b>unget</b>	Cancels a previous <b>get</b> command.
<b>val</b>	Validates Source Code Control System (SCCS) files.
<b>what</b>	Displays identifying information in files.

## Managing Object Files

<b>ar</b>	Maintains portable libraries used by the linkage editor.
<b>as</b>	Assembles a source file.
<b>dump</b>	Dumps selected parts of an object file.
<b>ld</b>	Links object files.
<b>lorder</b>	Finds the best order for member files in an object library.
<b>make</b>	Maintains up-to-date versions of programs.
<b>prof</b>	Displays program profile data.
<b>nm</b>	Displays the symbol table of an object file.
<b>size</b>	Displays the section sizes of common object files.
<b>strip</b>	Removes symbol and line number information from a common object file.
<b>touch</b>	Updates the access and modification times of a file.
<b>tsort</b>	Sorts an unordered list of ordered pairs (a topological sort).

---

## Playing Games

<b>arithmetic</b>	Tests arithmetic skills.
<b>back</b>	Plays backgammon.
<b>bj</b>	Plays blackjack.
<b>craps</b>	Plays craps.
<b>fish</b>	Plays the card game Go Fish.
<b>fortune</b>	Tells a fortune.
<b>hangman</b>	Plays hangman, the word-guessing game.
<b>moo</b>	Plays a number-guessing game.
<b>number</b>	Displays the written form of a number.
<b>quiz</b>	Tests your knowledge.
<b>ttt</b>	Plays tic-tac-toe.
<b>turnon</b>	Turns on execute permission for games.
<b>wump</b>	Plays the game Hunt the Wumpus.

---

## Commands



## acct/\*

## Purpose

Provides accounting shell procedures.

## Syntax

`/usr/lib/acct/chargefee` — *user* — *number* —|

`/usr/lib/acct/ckpacct` — { *1000* } —|  
 { *numblocks* } —|

`/usr/lib/acct/dodisk` — { *-o* } —| { *file* } —|

`/usr/lib/acct/lastlogin` —|

`/usr/lib/acct/monacct` — { *number*<sup>1</sup> } —|

`/usr/lib/acct/nulladm` — { *file* } —|

`/usr/lib/acct/prctmp` —|

`/usr/lib/acct/prdaily` — { *-l* } —| { *mmdd*<sup>2</sup> } —|  
 { *-c* } —|

<sup>1</sup> The default *number* is the current month.

<sup>2</sup> The default *mmdd* is the current day.

OL805236

OL805237



`/usr/lib/acct/prtacct` — 

<code>-f <i>fieldspec</i></code>
<code>-v</code>

 — 

<code>'<i>heading</i>'</code>
-------------------------------

 —

`/usr/lib/acct/remove`

`/usr/lib/acct/shutacct` — 

<code>'<i>reason</i>'</code>
------------------------------

 —

`/usr/lib/acct/startup` —

`/usr/lib/acct/turnacct` — 

one of
on
off
switch

 —

OL805238

## Description

**Note:** You should not share accounting files among nodes in a Distributed Services system. Each node should have its own copy of the various accounting files.

### chargefee

The **chargefee** command charges the specified *number* of units to the specified *user*. *number* can have an integer or decimal value. It writes a record to `/usr/adm/fee`, to be merged with other accounting records by the **runacct** command.

### ckpacct

The **ckpacct** command checks the size of `/usr/adm/pacct`. If the size exceeds the number specified in *numblocks*, **ckpacct** invokes **turnacct switch**. (The default value for *numblocks* is 1000.) If the number of free disk blocks in the `/usr` file system falls below 500, **ckpacct** automatically turns off the collection of process accounting records by invoking **turnacct off**. When 500 blocks are again available, accounting is activated again. This feature is sensitive to how frequently **ckpacct** is run (usually by **cron**).

### dodisk

The **dodisk** command performs the disk-usage accounting functions. **cron** normally runs this command periodically. By default, it does disk accounting on the special files whose stanzas in `/etc/filesystems` contain the attribute **account=true**. If you specify the **-o** flag, it does a slower version of disk accounting by login directory.

The *file* parameter specifies the one or more file system names where disk accounting is to be done. If you specify any file names, disk accounting is done on only these file systems. If you do not specify **-o**, *file* names should be the special file names of mountable file

---

**acct/\***


---

## Purpose

Provides accounting shell procedures.

## Syntax

`/usr/lib/acct/chargefee` *user* *number*

`/usr/lib/acct/ckpacct` *numblocks* *1000*

`/usr/lib/acct/dodisk` *-o* *file*

`/usr/lib/acct/lastlogin`

`/usr/lib/acct/monacct` *number*<sup>1</sup>

`/usr/lib/acct/nulladm` *file*

`/usr/lib/acct/prctmp`

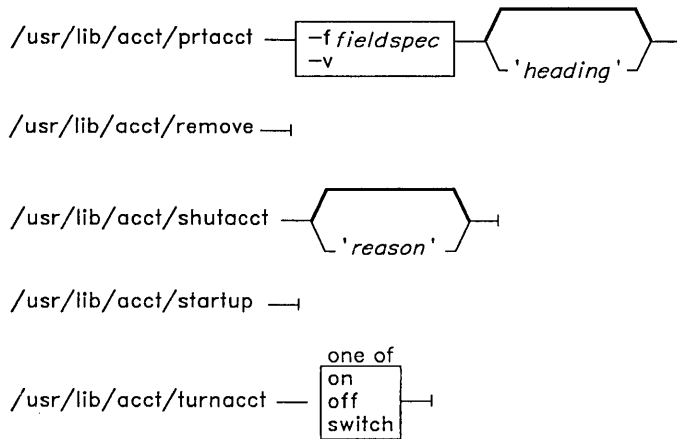
`/usr/lib/acct/prdaily` *-l* *mmd*<sup>2</sup> *-c*

<sup>1</sup> The default *number* is the current month.

<sup>2</sup> The default *mmd* is the current day.

OL805236

OL805237



OL805238

## Description

### chargefee

The **chargefee** command charges the specified *number* of units to the specified *user*. *number* can have an integer or decimal value. It writes a record to `/usr/adm/fee`, to be merged with other accounting records by the **runacct** command.

### ckpacct

The **ckpacct** command checks the size of `/usr/adm/pacct`. If the size exceeds the number specified in *numblocks*, **ckpacct** invokes **turnacct switch**. (The default value for *numblocks* is 1000.)

If the number of free disk blocks in the `/usr` file system falls below 500, **ckpacct** automatically turns off the collection of process accounting records by invoking **turnacct off**. When 500 blocks are again available, accounting is activated again. This feature is sensitive to how frequently **ckpacct** is run (usually by **cron**).

### dodisk

The **dodisk** command performs the disk-usage accounting functions. **cron** normally runs this command periodically. By default, it does disk accounting on the special files whose stanzas in `/etc/filesystems` contain the attribute **account=true**. If you specify the **-o** flag, it does a slower version of disk accounting by login directory.

The *file* parameter specifies the one or more file system names where disk accounting is to be done. If you specify any file names, disk accounting is done on only these file systems. If you do not specify **-o**, *file* names should be the special file names of mountable file

---

systems. If you specify both **-o** and *file* names, the files should be mount points of mounted file systems.

### **lastlogin**

The **lastlogin** command updates the file **/usr/adm/acct/sum/loginlog** to show the last date each user logged on. **runacct** normally calls this command.

### **monacct**

The **monacct** command performs monthly (or periodic) accounting. **cron** should run this command once each month or accounting period. *number* indicates the month or period to process. The default *number* is the current month. This default is useful if **monacct** is run by **cron** on the first day of each month. The **monacct** command creates summary files in **/usr/adm/acct/fiscal** and restarts summary files in **/usr/adm/acct/sum**.

### **nulladm**

The **nulladm** command creates *file*, assigns it permission code 664, and ensures that its owner and group are **adm**. (See “**chmod**” on page 128 for an explanation of file permissions.) Various accounting shell procedures call **nulladm**.

### **prctmp**

The **prctmp** command displays the session record file created by the **acctcon1** command (normally **/usr/adm/acct/nite/ctmp**).

### **prdaily**

The **prdaily** command formats a report of the day’s accounting data. Use *mmdd* to specify a date other than the current day. The report resides in **/usr/adm/acct/sum/rprtmmdd** where *mmdd* specifies the month and day of the report. **runacct** invokes this command to format a report of the previous day’s accounting data.

### **Flags**

- c** Reports exceptional resource usage by command, and may be used on the current day’s accounting data only.
- l** Reports exceptional usage by login ID for the specified date.

Daily reports are deleted (and thus inaccessible) each time **monacct** runs.

### **prtacct**

The **prtacct** command formats and displays any total accounting (**tacct**) file. You can specify a *heading* for the report by enclosing it in " " (double quotation marks).

### **Flags**

- ffieldspec** Selects fields to be displayed, using the field selection mechanism of **acctmerg**.
- v** Produces verbose output in which more precise notation is used for floating-point numbers.

### **remove**

The **remove** command deletes all **/usr/adm/acct/sum/wtmp\***, **/usr/adm/acct/sum/pacct\***, and **/usr/adm/acct/nite/lock\*** files.

### **shutacct**

The **shutacct** command turns process accounting off and adds a "reason" record to **/usr/adm/wtmp**. It is usually invoked during a system shutdown.

### **startup**

The **startup** command turns on the accounting functions when the system is started up. It should be called by the **/etc/rc** command file.

### **turnacct**

The **turnacct** command provides an interface to **accton** for turning process accounting **on** or **off**.

The **switch** flag turns accounting off, moves the current **/usr/adm/pacct** to the next free name in **/usr/adm/pacctincr**, where *incr* is a number starting at 1 and increased by one for each additional **pacct** file. After moving the **pacct** file, **turnacct** turns accounting back on.

This command is usually called by **ckpacct**, which in turn is called by **cron**, keeping the **pacct** file down to a manageable size.

## Files

<b>/usr/adm/fee</b>	Accumulator for fees charged to login names.
<b>/usr/adm/pacct</b>	Current file for process accounting.
<b>/usr/adm/pacct*</b>	Used if <b>pacct</b> gets large and during running of the daily accounting procedures.

---

/usr/adm/wtmp	Login/logout history file.
/usr/lib/acct/ptelus.awk	Shell procedure that calculates the limits for exceptional usage by login ID.
/usr/lib/acct/ptecms.awk	Shell procedure that calculates the limits of exceptional usage by command name.
/usr/adm/acct/nite	Working directory.
/usr/lib/acct	Holds all accounting commands.
/usr/adm/acct/sum	Summary directory.

## Related Information

The following commands: “**acctcms**” on page 36, “**acctcom**” on page 38, “**acctcon**” on page 42, “**acctmerg**” on page 46, “**acctprc**” on page 48, “**chmod**” on page 128, “**cron**” on page 172, “**fwtmp**” on page 345, and “**runacct**” on page 606.

The **acct** system call and the **acct**, **utmp**, and **filesystems** files in *AIX Operating System Technical Reference*.

“Running System Accounting” in *Managing the AIX Operating System*.

## acctcms

---

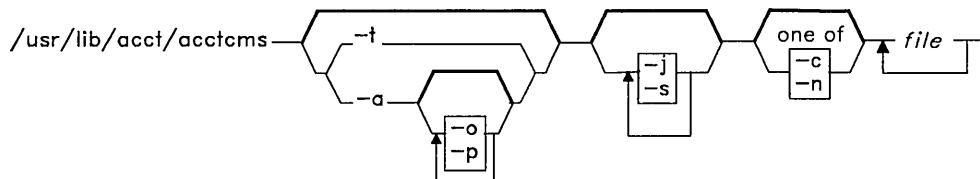
## acctcms

---

### Purpose

Produces command usage summaries from accounting records.

### Syntax



OL805421

### Description

The **acctcms** command reads the specified *files*. It adds together all records for identically named processes, sorts them, and writes them to standard output in a binary format. Files are usually in the **acct** file format described in *AIX Operating System Technical Reference*.

When you use the **-o** and **-p** flags together, **acctcms** produces a report that combines prime- and nonprime-time. All the output summaries are of total usage except for number of times run, CPU minutes, and real minutes, which are split into prime and nonprime minutes.

A typical sequence for performing daily command accounting and for maintaining a running total is:

```
acctcms file . . . > today
cp total previous total
acctcms -s today previous total > total
acctcms -a -s today
```

## Flags

- a Displays output in ASCII summary format rather than binary summary format. Each output line contains the command name, the number of times the command was run, its total *kcore-time*, its total *CPU time*, its total *real time*, its mean memory size (in K bytes), its mean CPU time per invocation of the command, and its *CPU usage factor*. The listed times are all in minutes. **acctcms** normally sorts its output by total kcore-minutes. The unit kcore-minutes measures the amount of storage used (in K-bytes) multiplied by the amount of time it was in use.
- c Sorts by total CPU time rather than total kcore-minutes.
- j Combines under the heading **\*\*\*other** all commands called only once.
- n Sorts by the number of times the commands were called.
- o Displays a command summary of nonprime-time commands only. You can use this flag with only the **-a** flag.
- p Displays a command summary of prime-time commands only. You can use this flag with only the **-a** flag.
- s Assumes that any named files that follow this flag are already in binary format.
- t Processes all records as total accounting records. The default binary format splits each field into prime and nonprime time sections.

## Related Information

The following commands: “**acct/\***” on page 31, “**acctcom**” on page 38, “**acctcon**” on page 42, “**acctmerg**” on page 46, “**acctprc**” on page 48, “**fwtmp**” on page 345, and “**runacct**” on page 606.

The **acct** system call and the **acct** and **utmp** files in *AIX Operating System Technical Reference*.

“Running System Accounting” in *Managing the AIX Operating System*.



# acctcom

---

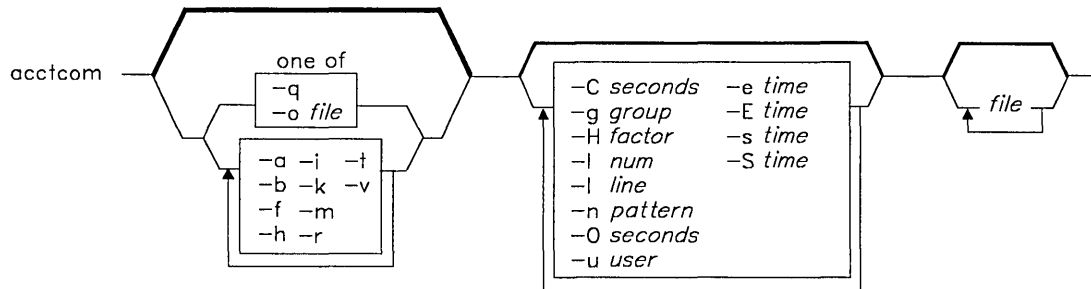
## acctcom

---

### Purpose

Displays selected process accounting record summaries.

### Syntax



OL805418

### Description

The **acctcom** command reads from specified *files*, from standard input, or from **/usr/adm/pacct** and writes records (selected by flags) to standard output. The input file format is described under **acct** in *AIX Operating System Technical Reference*.

If you do not specify any *file* parameters and if standard input is assigned to a work station or to **/dev/null** (as it is when a process runs in the background), **acctcom** reads **/usr/adm/pacct** instead of standard input.

By default, if you specify any *file* parameters, **acctcom** reads each chronologically by process completion time. Usually, **/usr/adm/pacct** is the current file that you want **acctcom** to examine. Because the **ckpacct** procedure keeps this file from growing too large, a busy system may have several **pacct** files. All but the current file have the following path name:

```
/usr/adm/pacct?
```

where ? is an integer incremented each time a new file is created.

Each record represents one completed process. The default display consists of the command name, user name, tty name, start time, end time, real seconds, CPU seconds, and mean memory size (in K bytes). These default items have the following headings in the output:

COMMAND NAME	USER	TTYNAME	START TIME	END TIME	REAL (SECS)	CPU (SECS)	MEAN SIZE(K)
-----------------	------	---------	---------------	-------------	----------------	---------------	-----------------

By using the appropriate flags, you can also display the fork/exec flag (F), the system exit value (STAT), the ratio of total CPU time to elapsed time (HOG FACTOR), the product of memory used and elapsed time (KCORE MIN), the ratio of user time to total (system and user) time (CPU FACTOR), the number of characters transferred in input/output operations (CHARS TRNSFD), and the total number of blocks read or written (BLOCKS READ).

If a process ran with superuser authority, its name is prefixed with a # (hash mark). If a process is not assigned to a known work station (for example, when **cron** runs it), a question mark (?) appears in the TTYNAME field.

**Note:** The **acctcom** command only reports on processes that have finished. Use the **ps** command to examine active processes.

If a specified time is later than the current time, it is interpreted as occurring on the previous day.

## Flags

- a Shows some average statistics about the processes selected. The statistics will be displayed after the output records.
- b Reads backwards, showing the most recent commands first. This flag has no effect when **acctcom** reads standard input.
- C *seconds* Shows only processes whose total CPU time (system time + user time), exceeds number of *seconds*.
- e *time* Selects processes existing at or before the specified time. You can use the **NLTIME** environment variable to specify the order of hours, minutes, and seconds. The default order is *hh[mm[ss]]*.
- E *time* Selects processes ending at or before the specified time. You can use the **NLTIME** environment variable to specify the order of hours, minutes, and seconds. The default order is *hh[mm[ss]]*. If you specify the same time for both the -E and -S flags, **acctcom** displays the process that existed at the specified time.
- f Displays the *fork/exec* flag and the system exit value columns in the output.
- g *group* Selects processes belonging to *group*. You can specify either the group ID or the group name.
- h Instead of mean memory size, shows the fraction of total available CPU time consumed by the process while it ran (**hog factor**). This factor is computed as:  
  
(total CPU time)/(elapsed time)

## acctcom

---

- H** *factor* Shows only processes that exceed *factor*. (See the **-h** flag for a discussion of how this factor is calculated.)
- i** Displays columns showing the number of characters transferred in read or write operations (the I/O counts).
- I** *num* Shows only processes transferring more than *num* characters.
- k** Instead of memory size, shows total kcore minutes.
- l** *line* Shows only processes belonging to work station */dev/line*.
- m** Shows mean main memory size. This flag is on by default. Specifying the **-h** or **-k** flags turns off **-m**.
- n** *pattern* Shows only commands matching *pattern*, where *pattern* is a regular expression like those in the **ed** command (see page 280), except that here you can use a **+** (plus sign) as a special symbol for one or more occurrences of the preceding character.
- o** *file* Copies selected process records to *file*, keeping the input data format. This flag suppresses writing to standard output.
- O** *seconds* Shows only processes with CPU system time exceeding *seconds*.
- q** Does not display any output records; just displays the average statistics that are displayed with the **-a** flag.
- r** Shows CPU factor. This factor is computed as:  
$$(\text{user-time}) / (\text{system-time} + \text{user-time}).$$
- s** *time* Shows only those processes that existed on or after the specified time. You can use the **NLTIME** environment variable to specify the order of hours, minutes, and seconds. The default order is *hh[mm[ss]]*.
- S** *time* Shows only those processes starting at or after the specified time. You can use the **NLTIME** environment variable to specify the order of hours, minutes, and seconds. The default order is *hh[mm[ss]]*.
- t** Shows separate system and user CPU times.
- u** *user* Shows only processes belonging to *user*. For *user*, you can give a user ID, a login name that is converted to a user ID, a **#** to select processes run with superuser authority, or a **?** to select processes associated with unknown user IDs.
- v** Eliminates column headings from the output.

## Files

/usr/adm/pacct	Current process accounting file.
/etc/passwd	User names and user IDs.
/etc/group	Group names and group IDs.

## Related Information

The following commands: “**acctdisk**” on page 44, “**acctcms**” on page 36, “**acctcon**” on page 42, “**acctmerg**” on page 46, “**acctprc**” on page 48, “**acct/\***” on page 31, “**fwtmp**” on page 345, “**ps**” on page 579, “**runacct**” on page 606, and “**su**” on page 724.

The **acct** system call, the **acct** and **utmp** files and the **environment** miscellaneous facility in *AIX Operating System Technical Reference*.

“Running System Accounting” and “Overview of International Character Support” in *Managing the AIX Operating System*.

## acctcon

---

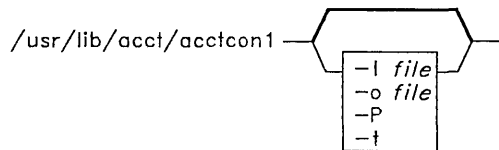
## acctcon

---

### Purpose

Performs connect-time accounting.

### Syntax



```
/usr/lib/acct/acctcon2 —
```

OL805233

### Description

#### acctcon1

The **acctcon1** command converts a sequence of login and logout records (read from standard input) to a sequence of login session records (written to standard output). its input should normally be redirected from **/usr/adm/wtmp**.

The **acctcon1** command displays, in ASCII format, the login device, user ID, login name, **prime connect time** (seconds), **nonprime connect time** (seconds), session starting time (numeric), and starting date and time (in date/time format). It also maintains a list of ports on which users are logged in. When it reaches the end of its input, it writes a session record for each port that still appears to be active. It normally assumes that its input is a current file, so that it uses the current time as the ending time for each session still in progress (see the **-t** flag on page 43).

#### Flags

**-l file** Writes to *file* a line-usage summary showing the line name, the number of minutes used, the percentage of total elapsed time used, the number of sessions charged, the number of logins, and the number of logouts. This file helps track line usage and identify bad lines. All hang-ups, terminations of **login**, and terminations of the login shell cause the system to write logout records, so the number of logouts is often much higher than the number of sessions.

- o *file* Writes to *file* an overall record for the accounting period, giving starting time, ending time, number of restarts, and number of date changes.
- p Displays input only, showing line name, login name, and time in both numeric and date/time formats.
- t Uses the last time found in the input as the ending time for any current processes instead of the current time. This is necessary in order to have reasonable and repeatable values for noncurrent files.

## acctcon2

The **acctcon2** command converts a sequence of login session records, produced by the **acctcon1** command, into total accounting records.

## Files

/usr/adm/wtmp      Login/logout history file.

## Related Information

The following commands: “**acdisk**” on page 44, “**acctcms**” on page 36, “**acctcom**” on page 38, “**acctmerg**” on page 46, “**acctprc**” on page 48, “**acct/\***” on page 31, “**fwtmp**” on page 345, “**init**” on page 396, “**login**” on page 453, and “**runacct**” on page 606.

The **acct** system call and the **acct** and **utmp** files in *AIX Operating System Technical Reference*.

“Running System Accounting” in *Managing the AIX Operating System*.

# acctdisk

---

## acctdisk

---

### Purpose

Performs disk-usage accounting.

### Syntax

```
/usr/lib/acct/acctdisk ->
/usr/lib/acct/acctdusg -u file -p /etc/passwd -p file ->
```

OL805192

### Description

#### acctdisk

The **acctdisk** command reads lines from standard input that contain a user ID, the user's login name, and the number of disk blocks occupied by his files. It converts these lines to total accounting records that can be merged with other accounting records and writes those records to standard output.

#### acctdusg

The **acctdusg** command reads a list of file names from standard input (usually piped from a **find / -print** command), computes disk resource usage (including **indirect blocks**) using the login name of the owner of the files, and writes the results to standard output.

#### Flags

- p file** Searches *file* for login names and numbers, instead of searching */etc/passwd*.
- u file** Places in *file* records of file names for which it does not charge.

### Files

<i>/etc/passwd</i>	Used to convert login names to user IDs.
<i>/usr/lib/acct</i>	Directory holding all accounting commands.

## Related Information

The following commands: “**acct/\***” on page 31, “**acctcms**” on page 36, “**acctcom**” on page 38, “**acctcon**” on page 42, “**acctmerg**” on page 46, “**acctprc**” on page 48, “**fwtmp**” on page 345, and “**runacct**” on page 606.

The **acct** system call and the **acct** and **utmp** files in *AIX Operating System Technical Reference*.

“Running System Accounting” in *Managing the AIX Operating System*.



# acctmerg

---

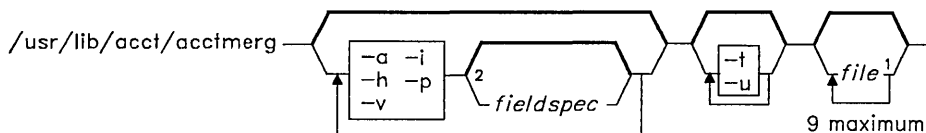
## acctmerg

---

### Purpose

Merges total accounting files.

### Syntax



<sup>1</sup>`acctmerg` always read standard input in addition to any named *files*.

<sup>2</sup>Do not put a blank between these items.

OL805234

### Description

The **acctmerg** command reads records from standard input and up to nine additional *files*, all in the **tacct** binary format or the **tacct** ASCII format. It merges these by adding records with keys (normally user ID and name) that are identical, and expects the input records to be sorted by those key fields. It writes these merged records to standard output.

The optional *fieldspecs* allow you to select input or output fields. A field specification is a comma-separated list of fields or field ranges. Field numbers are in the order specified in the **tacct** file in *AIX Operating System Technical Reference*, with array sizes, except for the *ta\_name* characters, taken into account. For example, `-h2-3,7,15-13,2` displays the login name, prime CPU and connect times, fee, queueing system, and disk usage data, and the login name again, in that order, with column headings. The default specification is "all fields" (`1-18` or `1-`), which produces very wide output lines containing all the available accounting data.

Queueing system, disk usage, or fee data can be converted into **tacct** records using the *-ifieldspec* argument. For example, disk accounting records, produced by **acctdisk**, consist of lines containing the user ID, login name, number of blocks, and number of disk samples (always 1). A file, **dacct**, containing these records can be merged into an existing total accounting file, **tacct**, with:

```
acctmerg -i1-2,13,18 <dacct |. acctmerg tacct >output
```

## Flags

- a**[fieldspec] Produces output in the form of ASCII records.
- h**[fieldspec] Displays column headings. This flag implies **-a** but is effective with **-p** or **-v**.
- i**[fieldspec] Expects input files composed of ASCII records.
- p**[fieldspec] Displays input without processing.
- t** Produces a single record that contains the totals of all input.
- u** Summarizes by user ID rather than by user name.
- v**[fieldspec] Produces output in ASCII format, with more precise notation for floating-point numbers.

## Example

The following sequence is useful for making repairs to any file in **tacct** format:

```
acctmerg -v <file1 >file2
        edit file2 as desired . . .
acctmerg -a <file2 >file1
```

## Related Information

The following commands: “**acct/\***” on page 31, “**acctcms**” on page 36, “**acctcom**” on page 38, “**acctcon**” on page 42, “**acctdisk**” on page 44, “**fwtmp**” on page 345, “**acctprc**” on page 48, and “**runacct**” on page 606.

The **acct** system call and the **acct** and **utmp** files in *AIX Operating System Technical Reference*.

“Running System Accounting” in *Managing the AIX Operating System*.

## acctprc

---

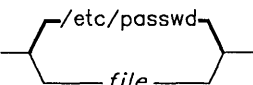
## acctprc

---

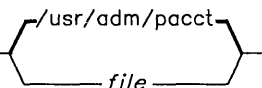
### Purpose

Performs process accounting.

### Syntax

`/usr/lib/acct/acctprc1`  `/etc/passwd`  
`file`

`/usr/lib/acct/acctprc2` 

`/usr/lib/acct/accton`  `/usr/adm/pacct`  
`file`

OL805235

### Description

#### **acctprc1**

The **acctprc1** command reads records from standard input that are in the **acct** format (described in *AIX Operating System Technical Reference*), adds the login names that correspond to user IDs, and then writes an ASCII record to standard output. This record contains the user ID, login name, prime CPU time, nonprime CPU time, the total number of characters transferred (in 512-byte units), the total number of blocks read and written, and mean memory size (in 64-byte units) for each process.

If specified, *file* contains a list of login sessions in **utmp** format (described in *AIX Operating System Technical Reference*), sorted by user ID and login name. By default, **acctprc1** gets login names from the password file, `/etc/passwd`. The information in *file* helps distinguish among different login names that share the same user ID.

**acctprc2**

The **acctprc2** command reads (from standard input) the records written by **acctprc1**, summarizes them by user ID and name, and writes the sorted summaries to standard output as total accounting records.

**accton**

The **accton** command without arguments turns process accounting off. If you specify *file* (the name of an existing file), the kernel adds process accounting records to it (**/usr/adm/pacct** by default).

**Files**

*/etc/passwd, /usr/adm/pacct*

**Related Information**

The following commands: “**acct/\***” on page 31, “**acctdisk**” on page 44, “**acctcms**” on page 36, “**acctcom**” on page 38, “**acctcon**” on page 42, “**acctmerg**” on page 46, “**fwtmp**” on page 345, and “**runacct**” on page 606.

The **acct** system call and the **acct** and **utmp** files in *AIX Operating System Technical Reference*.

“Running System Accounting” in *Managing the AIX Operating System*.

## actman

---

## actman

---

### Purpose

Lets you interact with multiple virtual terminals.

### Syntax

actman —l

OL805323

### Description

The **actman** command is the *Activity Manager* for the AIX Operating System. It is normally run by the AIX logger in the same manner as any program listed in the `/etc/passwd` file. Once started by the logger, **actman** creates the initial shell (`/bin/sh`) and monitors the number of open virtual terminals until all have been closed. It then exits to the AIX `init` process. If you try to end the initial shell when other virtual terminals are still open, **actman** restarts the initial shell.

To take advantage of the multiple virtual terminal capability, use the **open** command (see page 541) to execute another shell in a separate virtual terminal.

**Note:** You must log out of each existing shell to end your login session.

You do not need an Activity Manager if you do not have virtual terminal capabilities. Thus if you do not log in from the local console, **actman** overlays itself with the initial shell.

### Related Information

The following command: “**open**” on page 541.

“Using Display Station Features” in *Using the AIX Operating System*.

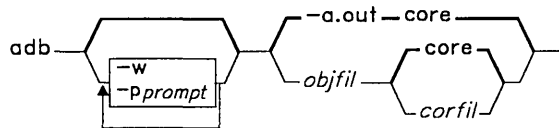
---

**adb**

---

**Purpose**

Provides a general purpose debugger.

**Syntax**

OL805465

**Description**

The **adb** command provides a debugger for C and assembler language programs. With it, you can examine object and core files and provide a controlled environment for running a program.

Normally, *objfil* is an executable program file that contains a symbol table. If *objfil* does not contain a symbol table, the symbolic features of **adb** cannot be used, although the file can still be examined. The default *objfil* is **a.out**.

The *corfil* is assumed to be a core image file produced by running *objfil*. The default *corfil* is **core**.

While running, **adb** takes input from standard input and writes to standard output. **adb** does not recognize the **Quit** or **Interrupt** keys. These keys cause **adb** to wait for a new command.

In general, requests to **adb** are of the form

```
[address] [,count] [command] [;]
```

where *address* and *count* are expressions. The default *count* is 1. If *address* is specified, then the expression **.** (dot) is set to *address*.

The interpretation of an address depends on the context it is used in. If a subprocess is being debugged, addresses are interpreted in the usual way in the address space of the subprocess. For more information, see "Addresses" on page 50.7.

You can enter more than one command at a time by separating the commands with a ; (semicolon).

**adb**

---

**Expressions**

- . Specifies the last address used by a command; this is also known as the current address.
- + Increases the value of . (dot) by the current increment.
- ^ Decreases the value of . (dot) by the current increment.
- " Specifies the last address typed by a command.
- integer* Specifies an octal number if *integer* begins with 0o, a hexadecimal number if preceded by 0x or #, a decimal number if preceded by 0t; otherwise, a number interpreted in the current radix. The radix is initially 16.
- '*cccc*' Specifies the ASCII value of up to 4 characters. \ (slash) can be used to escape a ' (apostrophe).
- < *name* Reads the current value of *name*. *name* is either a variable name or a register name. **adb** maintains a number of variables (see "Variables" on page 50.7) named by single letters or digits. If *name* is a register name, the value of the register is obtained from the system header in *corfil*. The register names are **r0...r15 pc ics cs mq**; the names **fp, pcp, and link** are recognized as synonyms for **r1, r14, and r15**.
- symbol* Specifies a sequence of upper or lower case letters, underscores, or digits, not starting with a digit. The value of the *symbol* is taken from the symbol table in *objfil*. An initial \_ (underscore) is prefixed to *symbol* if needed.
- symbol* Specifies, in C, the true name of an external symbol begins with \_ (underscore), as does the name of the constant pool of an external function. It may be necessary to use this name to distinguish it from internal or hidden variables of a program.
- .*symbol* Specifies the entry point of the function named by *symbol*.
- routine.name* Specifies the address of the variable *name* in the specified C routine. Both *routine* and *name* are *symbols*. If *name* is omitted the value is the address of the most recently activated C stack frame corresponding to *routine*.
- (*exp*) Specifies the value of the expression *exp*.

## Operators

Integers, symbols, variables, and register names can be combined with the following operators:

### Unary

<i>*exp</i>	Contents of location addressed by <i>exp</i> in <i>corefile</i> .
<i>@exp</i>	Contents of the location addressed by <i>exp</i> in <i>objfil</i> .
<i>-exp</i>	Integer negation.
<i>~exp</i>	Bitwise complement.

### Binary

<i>e1 + e2</i>	Integer addition.
<i>e1 - e2</i>	Integer subtraction.
<i>e1 * e2</i>	Integer multiplication.
<i>e1 % e2</i>	Integer division.
<i>e1 &amp; e2</i>	Bitwise conjunction.
<i>e1   e2</i>	Bitwise disjunction.
<i>e1 # e2</i>	<i>e1</i> rounded up to the next multiple of <i>e2</i> .

Binary operators are left associative and are less binding than unary operators.

## Commands

You can display the contents of a text or data segment with the *?* or the */* (slash) command. The *=* command displays a given address in the specified format *f*. (The commands *?* and */* may be followed by *\**, see “Addresses” on page 50.7.)

<i>?f</i>	Displays, in the format <i>f</i> , the contents of the <i>objfil</i> starting at <i>address</i> . The value of <i>.</i> (dot) increases by the sum of the increment for each format letter.
<i>/f</i>	Displays, in the format <i>f</i> , the contents of the <i>corfil</i> starting at <i>address</i> . The value of <i>.</i> (dot) increases by the sum of the increment for each <i>format</i> letter.
<i>=f</i>	Displays the value of <i>address</i> in the format <i>f</i> . The <i>i</i> and <i>s</i> format letters are not meaningful for this command.

The format consists of one or more characters that specify print style. Each format character may be preceded by a decimal integer that is a repeat count for the format character. While stepping through a format, *.* (dot) increments by the amount given for each format letter. If no format is given, the last format is used. The format letters available are as follows:

- o 2      Prints 2 bytes in octal.



**adb**

---

- O** 4 Prints 4 bytes in octal.
- q** 2 Prints 2 bytes in the current radix, unsigned.
- Q** 4 Prints 4 bytes in the current radix, unsigned.
- d** 2 Prints in decimal.
- D** 4 Prints long decimal.
- x** 2 Prints 2 bytes in hexadecimal.
- X** 4 Prints 4 bytes in hexadecimal.
- u** 2 Prints as an unsigned decimal number.
- U** 4 Prints long unsigned decimal.
- b** 1 Prints the addressed byte in the current radix, unsigned.
- c** 1 Prints the addressed character.
- C** 1 Prints the addressed character using the following escape conventions:
  1. Prints control characters as ~ followed by the corresponding printing character.
  2. Prints non-printable characters as ~ < *n* > where *n* is a hexadecimal value of the character. The character ~ prints as ~ ~.
- s** *n* Prints the addressed character until a zero character is reached.
- S** *n* Prints a string using the ~ escape convention. *n* specifies the length of the string including its zero terminator.
- Y** 4 Prints 4 bytes in date format (see **ctime** in *AIX Operating System Technical Reference*).
- i** *n* Prints as instructions. *n* is the number of bytes occupied by the instruction.
- a** 0 Prints the value of . (dot) in symbolic form. Symbols are checked to ensure that they have an appropriate type as indicated below.
  - / local or global data symbol
  - ? local or global text symbol
  - = local or global absolute symbol
- p** 4 Prints the addressed value in symbolic form using the same rules for symbol lookup as **a**.
- t** 0 When preceded by an integer, tabs to the next appropriate tab stop. For example, **8t** moves to the next 8-space tab stop.
- r** 0 Prints a space.
- n** 0 Prints a newline.

- "..." 0 Prints the enclosed string.
- ^ Decreases . (dot) by the current increment. Nothing prints.
- + Increases . (dot) by 1. Nothing prints.
- Decreases . (dot) decrements by 1. Nothing prints.

**newline**

Repeats the previous command incremented with a *count* of 1.

**[?/]lvalue mask**

Words starting at . (dot) are masked with *mask* and compared with *value* until a match is found. If **L** is used then the match is for 4 bytes at a time instead of 2. If no match is found then . (dot) is unchanged; otherwise . (dot) is set to the matched location. If *mask* is omitted then -1 is used.

**[?/]wvalue...**

Writes the 2-byte *value* into the addressed location. If the command is **W**, write 4 bytes. If the command is **V**, write 1 byte. Alignment restrictions may apply when using **w** or **W**.

**[?/]m b1 e1 f1[?/]**

Records new values for (*b1*, *e1*, *f1*). If less than three expressions are given then the remaining map parameters are left unchanged. If the ? or / is followed by \* then the second segment (*b2*, *e2*, *f2*) of the mapping is changed. If the list is terminated by ? or / then the file (*objfil* or *corfil* respectively) is used for subsequent requests. (For example, **/m?** causes / to refer to *objfil*).

**> name**

Assigns . (dot) to the variable or register *name*.

**!**

Calls a shell to read the rest of the line following !.

**\$modifier**

Miscellaneous commands. The available *modifiers* are:

- < *file* Reads commands from *file* and returns to the standard input.
- > *file* Sends output to *file*. If *file* is omitted, output returns to the standard output. *file* is created if it does not exist.
- r** Prints the general registers and the instruction addressed by **pc** and sets . (dot) to **pc**.
- b** Prints all breakpoints and their associated counts and commands.
- c** C stack backtrace. If *address* is given then it is taken as the address of the current frame (instead of using the frame pointer register). If **C** is used then the names and values of all automatic and static variables are printed for each active function. If *count* is given then only the first *count* frames are printed.

**adb**

---

- e Prints the names and values of external variables.
- w Sets the output page width for *address*. The default is 80.
- s Sets the limit for symbol matches to *address*. The default is 255.
- o Sets the current radix to 8.
- d Sets the current radix to *address* or 16, if none is specified.
- q Exits **adb**.
- v Prints all non-zero variables in octal.
- m Prints the address map.
- p Uses the remainder of the line as a prompt string.

*:modifier*

Manages a subprocess. Available *modifiers* are:

- bc Sets the breakpoint at *address*. The breakpoint runs *count* -1 times before causing a stop. Each time the breakpoint is encountered, the command *c* runs. If this command sets . (dot) to 0, the breakpoint causes a stop.
- d Deletes the breakpoint at *address*.
- r Runs *objfil* as a subprocess. If *address* is given explicitly, the program is entered at this point; otherwise, the program is entered at its standard entry point. *count* specifies how many breakpoints are to be ignored before stopping. Arguments to the subprocess may be supplied on the same line as the command. An argument starting with < or > causes the standard input or output to be established for the command. On entry to the subprocess, all signals are turned on.
- cs Continues the subprocess with signal *s* (see the **signal** system call in *AIX Operating System Technical Reference*). If *address* is given, the subprocess is continued at this address. If no signal is specified, the signal that caused the subprocess to stop is sent. Breakpoint skipping is the same as for **r**.
- ss Continues the subprocess in single steps *count* times. If there is no current subprocess, *objfil* is run as a subprocess as for **r**. In this case no signal can be sent; the remainder of the line is treated as arguments to the subprocess.
- k Stops the current subprocess, if one is running.

## Variables

**adb** provides a number of variables. On entry to **adb**, the following variables are set from the system header in the *corfil*. If *corfil* does not appear to be a **core** file, then these values are set from *objfil*.

<b>b</b>	The base address of the data segment
<b>d</b>	The size of the data segment
<b>e</b>	The entry address of the program
<b>m</b>	The “magic” number (0405, 0407, 0410, or 0411)
<b>s</b>	The size of the stack segment
<b>t</b>	The size of the text segment.

## Addresses

The address in a file associated with a written address is determined by a mapping associated with that file. Each mapping is represented by two triples (*b1*, *e1*, *f1*) and (*b2*, *e2*, *f2*). The *file address* that corresponds to a written *address* is calculated as follows:

$$b1 \leq \text{address} < e1 = > \text{file address} = \text{address} + f1 - b1$$

or

$$b2 \leq \text{address} < e2 = > \text{file address} = \text{address} + f2 - b2,$$

otherwise, the requested *address* is not legal. In some cases (for example, programs with separated I and D space) the two segments for a file may overlap. If a **?** or **/** is followed by an **\***, then only the second triple is used.

The initial setting of both mappings is suitable for normal **a.out** and **core** files. If either file is not of the kind expected, then for that file, *b1* is set to 0, *e1* is set to the maximum file size and *f1* is set to 0; in this way, the whole file can be examined with no address translation.

In order for **adb** to be used on large files, all appropriate values are kept as signed 32-bit integers.

## Flags

### **-pprompt**

Sets the prompt used by **adb** to *prompt*. If the prompt includes spaces, enclose the prompt in quotation marks.

**-w** Opens the *objfil* and *corfil* for writing. This flag makes either file if they do not exist.

## | **Files**

|           /dev/mem  
|           /dev/swap  
|           a.out  
|           core.

## | **Related Information**

|           The **ptrace** system call in *AIX Operating System Technical Reference*.  
|           The **a.out** and **core** files in *IBM RT PC AIX Operating System Technical Reference*.

---

# admin

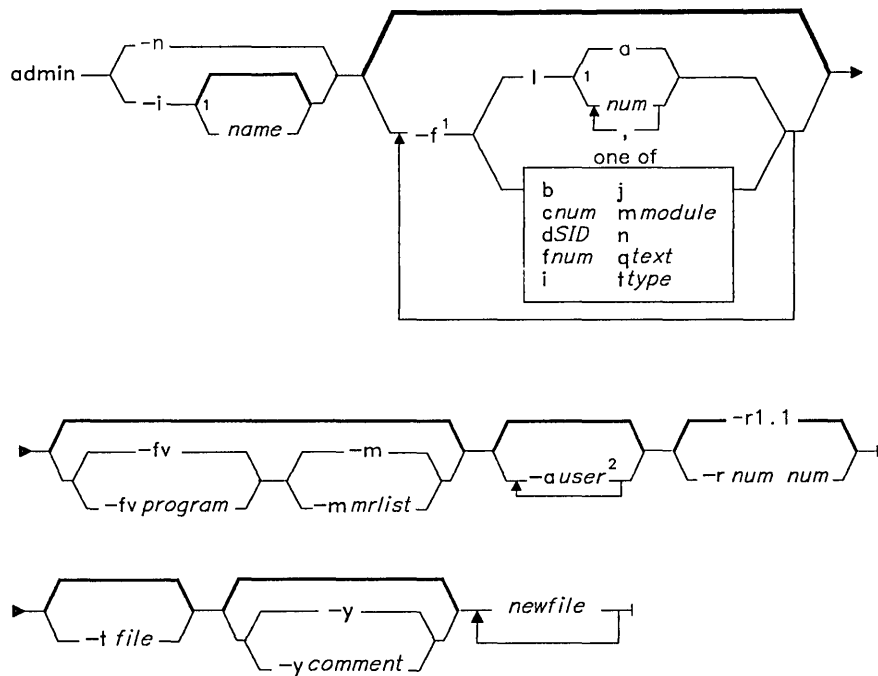
---

## Purpose

Creates and initializes SCCS files.

## Syntax

### To Create SCCS Files:



OL805376

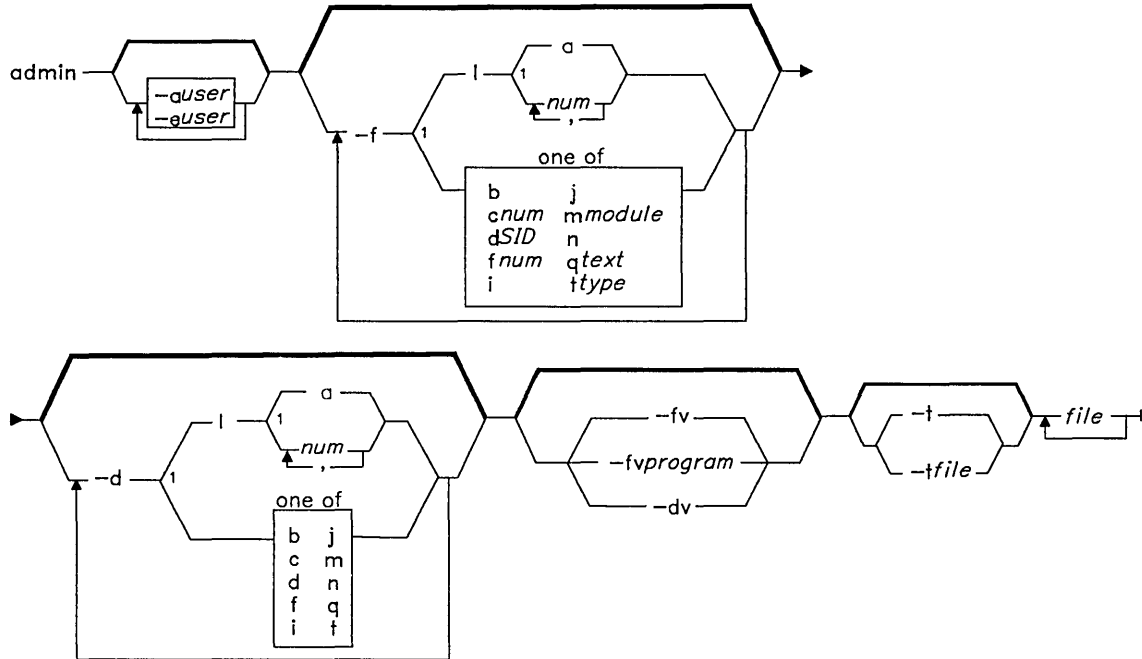
OL805160

<sup>2</sup> If `-a` is never used to specify users, then any user can run `get -e` on the file.

OL805417

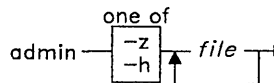
# admin

## To Change Existing SCCS Files:



OL805385

## To Check and Correct Damaged SCCS Files:



OL805158

<sup>1</sup> Do not put a blank between these items.

OL805308

## Description

The **admin** command creates new Source Code Control System (SCCS) files or changes specified parameters in existing SCCS files. These parameters control how the **get** command builds the files that you can edit. They also provide information about who can access the file, who can make changes, and when changes were made. The **admin** command is most often used to create new SCCS files without setting parameters. See

“Examples” on page 56 for the syntax used to create an SCCS file with no parameters set in the new file.

If the named *file* exists, **admin** modifies its parameters as specified by the flags. If it does not exist and you supply the **-i** or the **-n** flag, **admin** creates the new file and provides default values for unspecified flags. If you specify a directory name for *file*, **admin** performs the requested actions on all SCCS files in that directory (all files with the **s.** prefix). If you specify a **-** (minus) as a *file* name, **admin** reads standard input and interprets each line as the name of an SCCS file. An end-of-file character (**Ctrl-D**) ends input.

If you are not familiar with the delta numbering system, see *AIX Operating System Programming Tools and Interfaces* for more information.

## SCCS File Conventions

All SCCS file names must have the form **s.name**. New SCCS files are created with read-only permission. You must have write permission in the directory to create a file (see “**chmod**” on page 128 for an explanation of file permissions). **admin** writes to a temporary x-file, which it calls **x.name**. The x-file has the same permissions as the original SCCS file if it already exists, and it is read-only if **admin** creates a new file. After successful completion of **admin**, the x-file is moved to the name of the SCCS file. This ensures that changes are made to the SCCS file only if **admin** does not detect any errors while it is running.

Directories containing SCCS files should be created with permission code 755 (read, write, and execute permissions for owner, read and execute permissions for group members and others). SCCS files themselves should be created as read-only files (444). With these permissions, only the owner can use non-SCCS commands to modify SCCS files. If a group can access and modify the SCCS files then the directories should include group write permission.

The **admin** command also uses a temporary lock file (called **z.name**), to prevent simultaneous updates to the SCCS file by different users. See “SCCS Files” on page 360 for additional information on the **z.name** file.

The following table contains the header flags that can be set with the **-f** flag and unset with the **-d** flags (see page 55). The header flags control the format of the g-file created with the **get** command (see “SCCS Files” on page 360 for details on the g-file).



## admin

---

Header Flag	Header Flag Purpose
<b>b</b>	Lets you use the <b>-b</b> flag of a <b>get</b> command to create branch deltas.
<i>cnum</i>	Makes <i>num</i> the highest release number that a <b>get -e</b> can use. The value of <i>num</i> must be less than or equal to 9999. (Its default value is 9999.)
<i>fnum</i>	Makes <i>num</i> the lowest release number that a <b>get -e</b> can retrieve. <i>num</i> must be greater than 0 and less than 9999. (Its default value is 1.)
<i>dSID</i>	Makes <i>SID</i> the default delta supplied to a <b>get</b> command.
<b>i</b>	Treats the No id keywords ( <i>ge6</i> ) message issued by the <b>get</b> or <b>delta</b> command as an error (see “Identification Keywords” on page 362).
<b>j</b>	Permits concurrent <b>get</b> commands for editing the same SID of an SCCS file. This allows multiple concurrent updates to the same version of the SCCS file.
<i>lnum</i> [ <i>num</i> ] . . .	Locks the releases specified by <i>num</i> . . . against editing, so that a <b>get -e</b> against one of these releases fails. You can lock all releases against editing by specifying <b>-fla</b> and unlock specific releases with the <b>-d</b> flag.
<b>n</b>	Causes <b>delta</b> to create a null delta in any releases that are skipped when a delta is made in a new release. For example, if you make delta 5.1 after delta 2.7, releases 3 and 4 will be null. The resulting null deltas can serve as points from which to build branch deltas. Without this flag, skipped releases do not appear in the the SCCS file.
<i>qtext</i>	Substitutes <i>text</i> for all occurrences of the %Q% keyword in an SCCS text file retrieved by a <b>get</b> command. (See “Identification Keywords” on page 362 for more information on keywords.)
<i>mmodule</i>	Substitutes <i>module</i> for all occurrences of the %M% keyword in an SCCS text file retrieved by a <b>get</b> command. The default <i>module</i> is the name of the SCCS file without the <b>s.</b> prefix.
<i>ttype</i>	Substitutes <i>type</i> for all %Y% keywords in a g-file retrieved by a <b>get</b> .
<b>v</b> [ <i>program</i> ]	Makes <b>delta</b> prompt for Modification Request (MR) numbers as the reason for creating a delta. <i>program</i> specifies the name of an MR number validity checking program (see “ <b>delta</b> ” on page 236). If <b>v</b> is set in the SCCS file, the <b>admin -m</b> flag must also be used, even if its value is null.

Figure 1. SCCS Header Flags

## Flags

You can enter the flags and input file names in any order. All flags apply to all the files.

- 
- auser** Adds the specified *user* to the list of users that can make sets of changes (*deltas*), to the SCCS file. *user* can be either a user name, a group name, or a group ID. Specifying a group name or number is the same as specifying the names of all users in that group. You can specify more than one **-a** flag on a single **admin** command line. If an SCCS file contains an empty user list, then anyone can add deltas.
- If a file has a user list, the creator of the file must be included in the list in order for the creator to make deltas to the file.
- dhdrflag** Removes the specified header flag from the SCCS file. You can specify this flag only with existing SCCS files. You can also specify more than one **-d** flag in a single **admin** command. See Figure 1 on page 54 for the header flags that **admin** recognizes.
- euser** Removes the specified *user* from the list of users allowed to make deltas to the SCCS file. Specifying a group ID is equivalent to specifying all *user* names common to that group. You can specify several **-e** flags on a single **admin** command line.
- fhdrflag[*value*]** Places the specified header flag and value in the SCCS file. You can specify more than one header flag in a single **admin** command. See Figure 1 on page 54 for the header flags that **admin** recognizes.
- h** Checks the structure of the SCCS file and compares a newly computed checksum with the checksum that is stored in the first line of the SCCS file. When the checksum value is not correct, the file has been improperly modified or has been damaged. This flag helps you detect damage caused by the improper use of nonSCCS commands to modify SCCS files, as well as accidental damage. The **-h** flag prevents writing to the file, so it cancels the effect of any other flags supplied. If an error message is returned indicating the file is damaged, use the **-z** flag to recompute the checksum. Then test to see if the file is corrected by using the **-h** flag again.
- i[*name*]** Gets the text for a new SCCS file from *name*. This text is the first delta of the file. If you specify the **-i** flag but you omit the file name, **admin** reads the text from standard input until it reaches END OF FILE (Ctrl-D). If you do not specify the **-i** flag, but you do specify the **-n** flag, **admin** creates an empty SCCS file. **admin** can only create one file containing text at a time. If you are creating two or more SCCS files with one call to **admin**, you must use the **-n** flag, and the SCCS files created are empty.
- m[*mrlist*]** Specifies a list of Modification Requests (MR) numbers to be inserted into the SCCS file as the reason for creating the initial delta. The **v** flag must be set. The MR numbers are validated if the **v** flag has a value (the name of an MR number validation program). **admin** reports an error if the **v** flag is not set or if MR validation fails.

## admin

---

- n** Creates a new, empty SCCS file. Do not specify this flag when you use the **-i** flag.
- rnum.num** Inserts the initial delta into *num.num*, the release and version respectively. You can specify **-r** only if you also specify the **-i** or **-n** flag. If you do not specify this flag, the initial delta becomes release 1, version 1. Use this flag only when creating an SCCS file.
- t[file]** Takes descriptive text for the SCCS file from *file*. If you use **-t** when creating a new SCCS file, you must supply a file name. In the case of existing SCCS files:
- Without a file name, **-t** causes removal of the descriptive text (if any) currently in the SCCS file.
  - With a file name, **-t** causes text in the named file to replace the descriptive text (if any) currently in the SCCS file.
- y[comment]** Inserts *comment* text into the initial delta in a manner identical to that of the **delta** command. Use this flag only when you create an SCCS file. If you do not specify a comment, **admin** inserts a line of the following form:
- date and time created *YY/MM/DD HH:MM:SS* by login
- z** Recomputes the SCCS file checksum and stores it in the first line of the SCCS file (see the **-h** flag on page 55).
- Warning:** Using **admin** with this flag on a damaged file can prevent future detection of the damage. This flag should only be used if the SCCS file is changed using non-SCCS commands because of a serious error.

## Examples

1. To create an empty SCCS file named **s.prog.c**:  
admin -n s.prog.c

2. To convert an existing text file into an SCCS file:

```
admin -iprogram.c s.prog.c
```

This converts the text file `program.c` into the SCCS file `s.prog.c`. The original file remains intact, but it is no longer needed. You must rename or delete it before you can use the `get` command on `s.prog.c`.

## Related Information

The following commands: “**delta**” on page 236, “**ed**” on page 280, “**get**” on page 359, “**help**” on page 391, “**prs**” on page 574, and “**what**” on page 848.

The `sccsfile` file in *AIX Operating System Technical Reference*.

“Maintaining Different Versions of a Program” in *AIX Operating System Programming Tools and Interfaces*.

**ar**

---

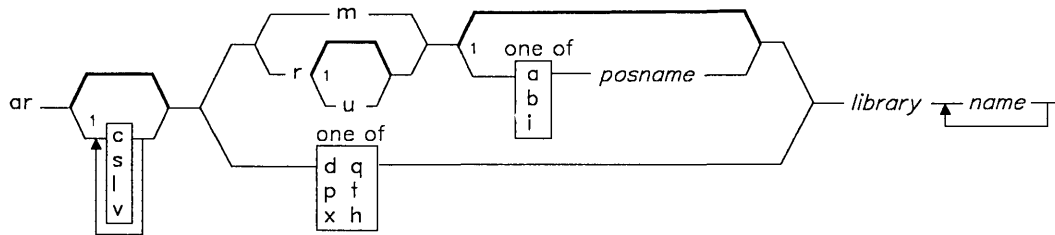
**ar**

---

## Purpose

Maintains portable libraries used by the linkage editor.

## Syntax



OL805377

`ar -w - library`

OL805349

<sup>1</sup> Do not put a blank between these items.

OL805308

## Description

The **ar** command combines one or more named files into a single *library* file written in **ar** archive format.

When **ar** creates a library, it creates headers in a transportable format; when it creates or updates one, it rebuilds the symbol table that the *linkage editor* (the **ld** command) uses to make efficient multiple passes over object file libraries. (The **ar** file entry in *AIX Operating System Technical Reference* describes the format and structure of portable archives and symbol tables.)

## Flags

**Note:** You must list all selected flags together on the command line, without blanks between them. You must always specify one from the set **dhmpqrtxw**. You can also specify any number of optional flags from the set **abcilsuv**. If you select a positioning flag

---

(**a**, **b**, or **i**), you must also specify the name of a file within *library* (*posname*), immediately following the flag list (separated from it by a blank).

- a** *posname* Positions the named files after the existing file identified by *posname*.
- b** *posname* Positions the named files before the existing file identified by *posname*.
- c** Suppresses the normal message that is produced when *library* is created.
- d** Deletes the named files from the library.
- h** Sets the modification times in the member headers of the named files to the current date and time. If you do not specify any file names, **ar** sets the time stamps of all member headers.
- i** *posname* Positions the named files before the existing file identified by *posname* (same as **b**).
- l** Places temporary files in the current (local) directory instead of directory **/tmp**.
- m** Moves the named files to some other position in the library. By default, it moves the named files to the end of the library. Use a positioning flag (**abi**) to specify some other position.
- p** Writes to the standard output the contents of the named *files* or all files in a *library* if you do not specify any files.
- q** Adds the named files to the end of the library. Positioning flags, if present, do not have any effect. Note that this process does not check to see if the named files are already in the library. In addition, if you name the same file twice, it may be put in the library twice.
- r** Replaces a named file if it already appears in the library. Since the named files occupy the same position in the library as the files they replace, a positioning flag does not have any additional effect. When used with the **u** flag (update), **r** replaces only files modified since they were last added to the library file.  
  
If a named file does not already appear in the library, **ar** adds it. In this case, positioning flags do affect placement. If you do not specify a position, new files are placed at the end of the library. If you name the same file twice, it may be put in the library twice.
- s** Forces the regeneration of the library symbol table whether or not **ar** modifies the library contents. Use this flag to restore the library symbol table after using the **strip** command on the library.
- t** Writes to the standard output a table of contents for the library. If you specify file names, only those files appear. If you do not specify any files, **t** lists all files in the library.

- u** Copies only files which have been changed since they were last copied (see the **r** flag discussed previously).
- v** Writes to standard output a verbose file-by-file description of the making of the new library. When used with the **t** flag, it gives a long listing similar to that of the **ls -l** command, described under “**ls**” on page 461. When used with the **x** flag, it precedes each file with a name. When used with the **h** flag, it lists the member name and the updated modification times.  
  
The environment variables **NLLDATE** and **NLTIME** control the format of the archive date and time.
- w** Displays the archive symbol table. Each symbol is listed with the name of the file in which the symbol is defined.
- x** Extracts the named files by copying them into the current directory. These copies have the same name as the original files, which remain in the library. If you do not specify any files, **x** copies all files out of the library. This process does not alter the library.

## Examples

1. To create a library:

```
ar vq lib.a strlen.o strcpy.o
```

If **lib.a** does not exist, then this creates it and enters into it copies of the files **strlen.o** and **strcpy.o**. If **lib.a** does exist, then this adds the new members to the end without checking for duplicate members. The **v** flag sets verbose mode, in which **ar** displays progress reports as it proceeds.

2. To list the table of contents of a library:

```
ar vt lib.a
```

This lists the table of contents of **lib.a**, displaying a long listing similar to **ls -l**. To list only the member file names, omit the **v** flag.

3. To replace or add new members to a library:

```
ar vr lib.a strlen.o strcat.o
```

This replaces the members **strlen.o** and **strcat.o**. If **lib.a** was created as shown in Example 1, then the **strlen.o** member is replaced. A member named **strcat.o** does not already exist, so it is added to the end of the library.

4. To specify where to insert a new member:

```
ar vrb strlen.o lib.a strcmp.o
```

This adds **strcmp.o**, placing the new member before **strlen.o**.

5. To update a member if it has been changed:

```
ar vru lib.a strcpy.o
```

This replaces the existing `strcpy.o` member, but only if the file `strcpy.o` has been modified since it was last added to the library.

6. To change the order of the library members:

```
ar vma strcmp.o lib.a strcat.o strcpy.o
```

This moves the members `strcat.o` and `strcpy.o` to positions immediately after `strcmp.o`. The relative order of `strcat.o` and `strcpy.o` is preserved. In other words, if `strcpy.o` preceded `strcat.o` before the move, then it still does.

7. To extract library members:

```
ar vx lib.a strcat.o strcpy.o
```

This copies the members `strcat.o` and `strcpy.o` into individual files named `strcat.o` and `strcpy.o`, respectively.

8. To extract and rename a member:

```
ar p lib.a strcpy.o >stringcopy.o
```

This copies the member `strcpy.o` to a file named `stringcopy.o`.

9. To delete a member:

```
ar vd lib.a strlen.o
```

This deletes the member `strlen.o` from the library `lib.a`.

## Files

`/tmp/ar*` Temporary files.

## Related Information

The following commands: “**backup**” on page 76, “**ld**” on page 427, “**lorder**” on page 457, “**make**” on page 474, “**nm**” on page 521, “**size**” on page 665, and “**strip**” on page 716.

The **a.out** and **ar** files and **environment** miscellaneous facility in *AIX Operating System Technical Reference*.

The “Overview of International Character Support” in *Managing the AIX Operating System*.



# arithmetic

---

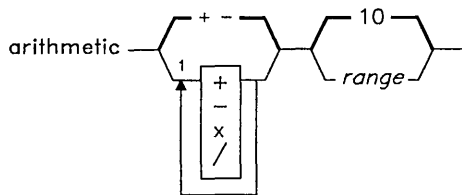
## arithmetic

---

### Purpose

Tests arithmetic skills.

### Syntax



<sup>1</sup> Do not put a blank between these items.

OL805164

OL805308

### Description

The **arithmetic** command displays simple arithmetic problems and waits for you to enter an answer. If your answer is correct, the program displays *Right!* and presents a new problem. If your answer is wrong, it displays *What?* and waits for another answer. Every 20 problems, **arithmetic** displays the number of correct and incorrect responses and the time required to answer.

The **arithmetic** command does not give the correct answers to the problems it displays. It provides practice rather than instruction in performing arithmetic calculations.

The *range* is a decimal number specifying the permissible range of all numbers (except answers). The default range is 10. At the start, all numbers within this range are equally likely to appear. If you make a mistake, the numbers in the problem you missed become more likely to reappear.

To quit the game, press **INTERRUPT (Alt-Pause)**; **arithmetic** displays the final game statistics and exits.

## Flags

Two types of optional flags modify the action of **arithmetic**. The first set specifies the type of arithmetic problem:

- + Specifies addition problems.
- Specifies subtraction problems.
- x Specifies multiplication problems.
- / Specifies division problems.

If you do not select any flags, **arithmetic** selects addition and subtraction problems. If you give more than one problem specifier (+-x/), the program mixes the specified types of problems in random order.

## Examples

1. To drill on addition and subtraction of integers from 0 to 10:  
`/usr/games/arithmetic`
2. To drill on addition, multiplication, and division of integers from 0 to 50:  
`/usr/games/arithmetic +x/ 50`

**as**

---

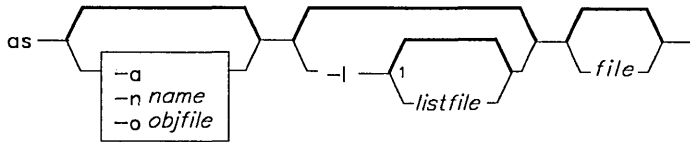
**as**

---

## Purpose

Assembles a source file.

## Syntax



OL805165

<sup>1</sup> Do not put a blank between these items.

OL805308

## Description

The **as** command reads and assembles the named *file* (conventionally this file ends with a *.s* suffix). If you do not specify a *file*, **as** reads and assembles standard input. It stores its output, by default, in a file named **a.out**. The output file is executable if no errors occur and if there are no unresolved external references.

## Flags

- a** Provide extended addressing for handling large structures.
- l[*listfile*]** Produces an assembler listing. If you do not specify a file name, a default name is produced by replacing the *.s* extension of the source file name with an *.lst* extension.
- n *name*** Specifies the name that appears in the header of the assembler listing. By default, the header contains the name of the assembler source file.
- o *objfile*** Writes the output of the assembly process to the specified file instead of to **a.out**.

## **Files**

a.out      Default output file.

## **Related Information**

The following commands: “**cc**” on page 112 and “**ld**” on page 427.

The **a.out** file in *AIX Operating System Technical Reference*.

The discussion of **as** in *Assembler Language Reference* and *AIX Operating System Programming Tools and Interfaces*.

# at

---

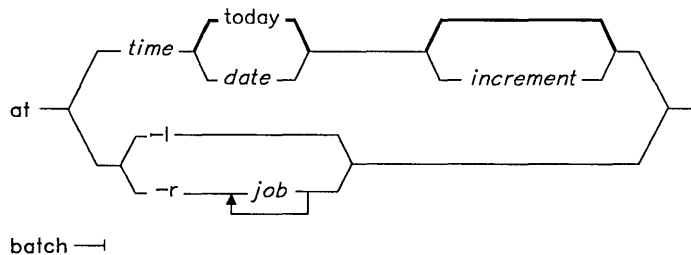
## at, batch

---

### Purpose

Runs commands at a later time.

### Syntax



OL805002

### Description

The **at** and **batch** commands read from standard input the names of commands to be run at a later time:

- **at** allows you to specify when the commands should be run.
- **batch** runs jobs when the system load level permits.

Both **at** and **batch** mail you all output from standard output and standard error for the scheduled commands, unless you redirect that output. They also write the job number and the scheduled time to standard error.

Variables in the shell environment, the current directory, **umask**, and **ulimit** are retained when the commands are run. Open file descriptors, traps, and priority are lost.

You can use **at** if your name appears in the file **/usr/lib/cron/at.allow**. If that file does not exist, **at** checks the file **/usr/lib/cron/at.deny** to determine if you should be denied access to **at**. If neither file exists, only the superuser can submit a job. The allow/deny files contain one user name per line. If **at.allow** does exist, the superuser's login name must be included in it for the superuser to be able to use the command.

The required *time* parameter can be one of the following:

1. A number followed by an optional suffix. **at** interprets one-and two-digit numbers as hours. It interprets four digits as hours and minutes. The **NLTIME** environment variable specifies the order of hours and minutes. The default order is the hour followed by the minute. You can also separate hours and minutes with a **:** (colon). The default order is *hour:minute*.

In addition, you may specify a suffix of **am**, **pm**, or **zulu**. If you do not specify **am** or **pm**, **at** uses a 24 hour clock. The suffix **zulu** indicates that the time is **GMT** (Greenwich Mean Time). The **NLTMISC** environment variable controls the suffixes that **at** recognizes.

2. **at** also recognizes the following keywords as special *times*: **noon**, **midnight**, and **now**. Note that you can use the special word **now** only if you also specify a *date* or an *increment*. Otherwise, **at** tells you: **too late**. The **NLTSTRS** environment variable controls the additional keywords that **at** recognizes.

You may specify the *date* parameter as either a month name and a day number (and possibly a year number preceded by a comma), or a day of the week. The **NLDATE** environment variable specifies the order of the month name and day number (by default, month followed by day). The **NLLDAY** environment variable specifies long day names; **NLSDAY** and **NLSMONTH** specify short day and month names. (By default, the long name is fully spelled out; the short name abbreviated to three characters.) **at** recognizes two special “days,” **today** and **tomorrow** by default. (The **NLTSTRS** environment variable specifies these special days.) **today** is the default *date* if the specified time is later than the current hour; **tomorrow** is the default if the time is earlier than the current hour. If the specified month is less than the current month (and a year is not given), next year is the default year. The optional *increment* can be one of the following:

1. A **+** (plus sign) followed by a number and one of the following words: **minute[s]**, **hour[s]**, **day[s]**, **week[s]**, **month[s]**, **year[s]** (or their non-English equivalents).
2. The special word **next** followed by one of the following words: **minute[s]**, **hour[s]**, **day[s]**, **week[s]**, **month[s]**, **year[s]** (or their non-English equivalents).

The **NLTUNITS** environment variable specifies the non-English equivalents of the English defaults.

## Flags

- l Reports your scheduled jobs.
- r *job* . . . Removes *jobs* previously scheduled by **at** or **batch**, where *job* is the number assigned by **at** or **batch**. If you do not have superuser authority (see “**su**” on page 724), you can remove only your own jobs.

## Examples

1. To schedule the command from the terminal, use a command similar to one of the following:

at 5 pm Friday uuclean Ctrl-D	at now next week uuclean Ctrl-D	at now + 2 days uuclean Ctrl-D
--	---------------------------------------	--------------------------------------

2. To run uuclean at 3:00 in the afternoon on the 24th of January, use any one of the following commands:

```
echo uuclean | at 3:00 pm January 24
echo uuclean | at 3pm Jan 24
echo uuclean | at 1500 jan 24
```

3. To run a job when the system load permits:

```
batch <<!
longjob 2>&l >outfile | mail myID
!
```

This example shows the use of a *here document* to send standard input to `at` (see “Inline Input Documents” on page 650).

The order of redirections is important here, so that only error messages are sent into the pipe to the `mail` command. If you reverse the order, both standard error and standard output are sent to `outfile` (see the discussion of “Input and Output Redirection Using File Descriptors” on page 651 for details).

4. To have a job reschedule itself, invoke `at` from within the shell procedure by including code similar to the following within the shell file:

```
echo "sh shellfile" | at now tomorrow
```

5. To list the jobs you have sent to be run later:

```
at -l
```

6. To cancel jobs:

```
at -r 103 227
```

This cancels jobs 103 and 227. Use `at -l` to list the job numbers assigned to your jobs.

## Files

/usr/lib/cron	Main cron directory.
/usr/lib/cron/at.allow	List of allowed users.
/usr/lib/cron/at.deny	List of denied users.
/usr/spool/cron/atjobs	Spool area.

## Related Information

The following commands: “**cron**” on page 172, “**kill**” on page 422, “**mail**” on page 470, “**nice**” on page 515, “**ps**” on page 579, and “**sh**” on page 637.

The **environment** special facility in *AIX Operating System Technical Reference*.

“Running Commands at Pre-set Times” and “Overview of International Character Support” in *IBM RT PC Managing the AIX Operating System*.



# awk

---

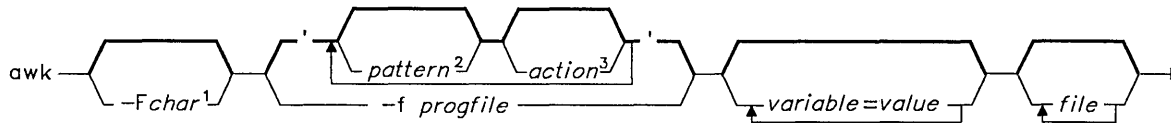
## awk

---

### Purpose

Finds lines in files matching specified patterns and performs specified actions on them.

### Syntax



<sup>1</sup> The default *char* is a tab.

<sup>2</sup> The default pattern is every line.

<sup>3</sup> The default action is to print the line.

OL805422

### Description

The **awk** command is a more powerful pattern matching command than the **grep** command. It can perform limited processing on the input lines, instead of simply displaying lines that match. Some of the features of **awk** are:

- It can perform convenient numeric processing.
- It allows variables within actions.
- It allows general selection of patterns.
- It allows control flow in the actions.
- It does not require any compiling of programs.

For a detailed discussion of **awk**, see *AIX Operating System Programming Tools and Interfaces*.

The **awk** command, reads *files* in the order stated on the command line. If you specify a file name as - (minus) or do not specify a file name, **awk** reads standard input.

The **awk** command searches its input line by line for *patterns*. When it finds a match, it performs the associated *action* and writes the result to standard output. Enclose *pattern-action* statements on the command line in single quotation marks to protect them from interpretation by the shell.

The **awk** command first reads all pattern-action statements, then it reads a line of input and compares it to each pattern, performing the associated actions on each match. When it has compared all patterns to the input line, it reads the next line.

The **awk** command treats input lines as fields separated by spaces, tabs, or a field separator you set with the **FS** variable. Fields are referenced as **\$1**, **\$2**, and so on. **\$0** refers to the entire line.

On the **awk** command line, you can assign *values* to variables as follows:

```
variable = value
```

## Pattern-Matching Statements

Pattern-matching statements follow the form:

```
pattern      { action }
```

If a *pattern* lacks a corresponding *action*, **awk** writes the entire line that contains the pattern to standard output. If an *action* lacks a corresponding *pattern*, it matches every line.

## Actions

An action is a sequence of statements that follow C Language syntax. These statements can include:

<b><i>statement</i></b>	<b><i>format</i></b>
<b>if</b>	<b>if ( <i>conditional</i> ) <i>statement</i> [ <b>else</b> <i>statement</i> ]</b>
<b>while</b>	<b>while ( <i>conditional</i> ) <i>statement</i></b>
<b>for</b>	<b>for ( <i>expression</i> ; <i>conditional</i> ; <i>expression</i> ) <i>statement</i></b>
<b>break</b>	
<b>continue</b>	
<b>{ <i>statement</i> . . . }</b>	
<b>(<i>assignment</i>)</b>	<b><i>variable</i> = <i>expression</i></b>
<b>print</b>	<b>print [<i>expression-list</i>] [&gt; <i>expression</i>]</b>
<b>printf</b>	<b>printf <i>format</i> [, <i>expression-list</i>] [&gt; <i>expression</i>]</b>
<b>next</b>	
<b>exit</b>	

Statements can end with a semicolon, a new-line character , or the right brace enclosing the action.

If you do not supply an action, **awk** displays the whole line. Expressions can have string or numeric values and are built using the operators **+**, **-**, **\***, **/**, **%**, a blank for string concatenation, and the C operators **++**, **--**, **+=**, **-=**, **\*=**, **/=**, and **%=**. In statements, variables may be scalars, array elements (denoted **x[i]**) or fields. Variable names may consist of upper- and lowercase alphabetic letters, the underscore character, the digits (0-9), and extended characters. Variable names cannot begin with a digit. Variables are

initialized to the null string. Array subscripts may be any string; they do not have to be numeric. This allows for a form of associative memory. String constants in expressions should be enclosed in double quotation marks.

There are several variables with special meaning to **awk**. They include:

<b>FS</b>	Input field separator (default is a blank). This separator character cannot be a two-byte extended character.
<b>NF</b>	The number of fields in the current input line (record).
<b>NR</b>	The number of the current input line (record).
<b>FILENAME</b>	The name of the current input file.
<b>OFS</b>	The output field separator (default is a blank). This separator character cannot be a two-byte extended character.
<b>ORS</b>	The output record separator (default is a new-line character). This separator character cannot be a two-byte extended character.
<b>OFMT</b>	The output format for numbers (default <code>%.6g</code> ).

Since the actions process fields, input white space is not preserved on the output.

The **printf** statement formats its expression list according to the format of the **printf** subroutine (see *AIX Operating System Technical Reference*), and writes its arguments to standard output, separated by the output field separator and terminated by the output record separator. You can redirect the output using the **print > file** or **printf > file** statements.

You have two ways to designate a character other than white space to separate fields. You can use the **-Fc** flag on the **awk** command line, or you can start *progfile* with:

```
BEGIN { FS = c }
```

Either action changes the field separator to *c*.

There are several built-in functions that can be used in **awk** actions.

<b>length</b>	Returns the length of the whole line if there is no argument or the length of its argument taken as a string.
<b>exp(<i>n</i>)</b>	Takes the exponential of its argument.
<b>log(<i>n</i>)</b>	Takes the base e logarithm of its argument.
<b>sqrt(<i>n</i>)</b>	Takes the square root of its argument.
<b>int(<i>n</i>)</b>	Takes the integer part of its argument.
<b>substr(<i>s,m,n</i>)</b>	Returns the substring <i>n</i> characters long of <i>s</i> , beginning at position <i>m</i> .
<b>sprintf(<i>fmt,expr,expr, . . .</i>)</b>	Formats the expressions according to the <b>printf</b> format string <i>fmt</i> and returns the resulting string.

## Patterns

Patterns are arbitrary Boolean combinations of patterns and relational expressions (the `!`, `||`, and `&&` operators and parentheses for grouping). You must start and end patterns with slashes (`/`). You can use regular expressions like those allowed by the `egrep` command (see “`grep`” on page 381), including the following special characters:

- `+` One or more occurrences of the pattern.
- `?` Zero or one occurrences of the pattern.
- `|` Either of two statements.
- `()` Grouping of expressions.

Isolated patterns in a pattern apply to the entire line. Patterns can occur in relational expressions. If two patterns are separated by a comma, the action is performed on all lines between an occurrence of the first pattern and the next occurrence of the second. Regular expressions can contain extended characters with one exception: range constructs in character class specifications using square brackets cannot contain two-byte extended characters. Individual instances of extended characters can appear within square brackets; however, two-byte extended characters are treated as two separate one-byte characters. Regular expressions can also occur in relational expressions.

There are two types of relational expressions that you can use. One has the form:

*expression matchop pattern*

where *matchop* is either: `~` (for “contains”) or `!~` (for “does not contain”). The second has the form:

*expression relop expression*

where *relop* is any of the six C relational operators: `<`, `>`, `<=`, `>=`, `==`, and `!=`. A conditional can be an arithmetic expression, a relational expression, or a Boolean combination of these.

You can use the special patterns **BEGIN** and **END** to capture control before the first and after the last input line is read, respectively. You can only use these patterns before the first and after the last line in *progfile*.

There are no explicit conversions between numbers and strings. To force an expression to be treated as a number, add `0` to it. To force it to be treated as a string, append a null string (`""`).

## Flags

- `-f progfile` Searches for the patterns and perform the actions found in the file *progfile*.
- `-Fchar` Uses *char* as the field separator character (by default a blank).

## Examples

1. To display the lines of a file that are longer than 72 characters:

```
awk "length >72" chapter1
```

This selects each line of the file `chapter1` that is longer than 72 characters. **awk** then writes these lines to standard output because no *action* is specified.

2. To display all lines between the words `start` and `stop`:

```
awk "/start/,/stop/" chapter1
```

3. To run an **awk** program (`sum2.awk .`) that processes a file (`chapter1`):

```
awk -f sum2.awk chapter1
```

The following **awk** program computes the sum and average of the numbers in the second column of the input file:

```
{
    sum += $2
}

END {
    print "Sum: ", sum;
    print "Average:", sum/NR;
}
```

The first action adds the value of the second field of each line to the variable `sum`. **awk** initializes `sum` (and all variables) to zero before starting. The keyword **END** before the second action causes **awk** to perform that action after all of the input file has been read. The variable **NR**, which is used to calculate the average, is a special variable containing the number of records (lines) that have been read.

4. To print the names of the users who have the C shell as the initial shell:

```
awk -F: '/csh/{print $1}' /etc/passwd
```

## Related Information

The following commands: “**lex**” on page 432, “**grep**” on page 381, and “**sed**” on page 629.

The **printf** subroutine in *AIX Operating System Technical Reference*.

The “Overview of International Character Support” in *Managing the AIX Operating System*.

The discussion of **awk** in *AIX Operating System Programming Tools and Interfaces*.

---

# back

---

## Purpose

Plays backgammon.

## Syntax

| /usr/games/back —|

OL805186

## Description

The **back** game provides you with a partner for backgammon. You select one of three skill levels: beginner, intermediate, or expert. You may also choose to roll your own dice during your turns, and you are asked if you want to move first.

The points are numbered such that:

- 0 is the bar for removed white pieces.
- 1 is white's extreme inner table.
- 24 is brown's extreme inner table.
- 25 is the bar for removed brown pieces.

For details on how to make your moves, enter *y* when **back** asks **Instructions** at the beginning of the game. When it first asks **Move?**, enter *?* to see a list of choices other than entering a numerical move.

When the game is finished, **back** asks you if you want to save game information. A *y* response stores game data in the file **back.log** in your current directory.

The **back** game plays only the forward game, even at the expert level. It will object if you try to make too many moves in a turn, but not if you make too few. Doubling is not implemented.

To quit the game, press **INTERRUPT (Alt-Pause)**.

## Files

/usr/games/lib/backrules	Rules file.
/tmp/b*	Log temp file.
back.log	Log file.

# backup

---

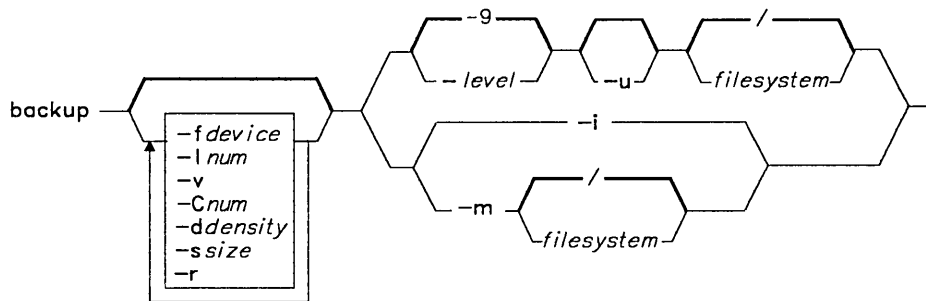
## backup

---

### Purpose

Backs up files.

### Syntax



OL805082

### Description

The **backup** command copies files in the **backup** format described in *AIX Operating System Technical Reference*. The backup output device is usually a removable medium, such as diskette or magnetic tape. You can name either a file system (**backup by i-node**) or the actual files to be backed up (**backup by name**). In the first case, you can back up either all files on the system (a **full backup**) or only the files that have been modified since a specific full backup (an **incremental backup**). You can also specify a minidisk (**backup by minidisk**), in which case **backup** copies an exact image of the entire minidisk.

**Note:** Because a backup by minidisk backs up an entire minidisk as an exact image, a large minidisk with a small or sparsely used file system may take longer and require more backup medium to back up this way, rather than by i-node or by name.

If the file system you are backing up is mounted and is not the root file system, **backup** unmounts the file system before it performs an i-node backup and then remounts the file system before quitting.

If the file systems you are backing up include the root file system, **backup** ensures that the other file systems are not in use. If one is, it warns you of this and quits.

For a file system backup, you supply a *level* number and a *filesystem* name. The possible level numbers are 0-9; the default level is 9. A level 0 backup includes all files on the file

system. A level *n* backup includes all files modified since the last level *n-1* backup. The level numbers, in conjunction with the **-u** flag, provide an easy way to maintain a hierarchy of incremental backups for each file system. See *Managing the AIX Operating System* for a discussions of backup strategy and the use of incremental backups.

The name of a *filesystem* can be either the physical device name (the block or raw name) or the name of the directory on which the file system is normally mounted. When you specify a directory name, **backup** reads **/etc/filesystems** for the physical device name. In this case, it also acquires values for other backup parameters from **/etc/filesystems**.

**Note:** The *filesystem* must specify a local device or directory.

By default, **backup** writes to the device defined in the **backupdev** entry of **/etc/filesystems** for a backup by i-node or to **/dev/rfd0** for backups by minidisk or name or if no **backupdev** is defined in **/etc/filesystems**. You can override this default action with the **-f** flag. The **backup** command recognizes a special syntax for the names of output files. If the argument is a range of names, such as **/dev/rfd0-3**, **backup** automatically goes from one drive (in the range) to the next. After exhausting all of the specified drives, it halts and requests that new volumes be mounted.

For individual file backup, use the **-i** flag. **backup** reads standard input for the names of files to be backed up. In this case, **backup** does not read **/etc/filesystems** and does not default to the settings specified there.

## Flags

**Warning:** Ensure that the flags you specify match the backup medium. If the backup medium is not a disk or diskette, do not specify the **-I** flag. Similarly, if the backup medium is not a tape, do not specify the **-d** or **-s** flags. If you do specify flags that do not go with the medium, **backup** displays an appropriate error message and continues the backup.

- Cnum** Specifies the number of blocks to write in a single output operation. If you do not specify *num*, **backup** uses a default value appropriate for the physical device selected. Larger values of *num* result in longer physical transfers to tape devices. The value of the **-C** flag is always ignored when **backup** writes to diskette. In this case, it always writes in clusters that occupy a complete track.
- ddensity** Specifies the *density* of a tape medium in bytes per inch. The default density is 700 bytes per inch.
- fdevice** Specifies the output device.
- i** Reads standard input for the names of files to back up.
- lnum** Uses *num* as the limit of the total number of block to use on a diskette. The default value is the entire diskette (2400 blocks).



## backup

---

- m** Backs up the entire minidisk as an exact image.
- Note:** Incremental backups are not supported for this mode of backup.
- r** Indicates that removable medium is ready to use. When you specify this flag, **backup** proceeds without prompting you to prepare the backup medium or waiting for you to press the **Enter** key to continue.
- slength** Specifies the *length* in feet of usable space on a tape medium. This is a combination of the physical length and the number of tracks on the tape. In the case of IBM RT PC Streaming Tape, you should multiply the physical length of the tape by 9 (the number of tracks) to determine the usable space available.
- u** Updates the time, date, and level of the backup in the `/etc/budate` file. This file provides the information needed for incremental backups.
- v** Reports on each phase of the backup as it is completed and gives regular progress reports during the longest phase.
- level** Specifies the backup *level* (0-9). The default *level* is 9.
- You should use the **-u** flag when you do an incremental backup to ensure that information regarding the last date, time, and level of each incremental backup is written to the file `/etc/budate`.

## Examples

1. To back up an entire file system:

```
backup -0 -u /
```

This backs up the entire (-0) root file system (/) to the device defined in the **backupdev** entry in `/etc/filesystems`. It also updates the current backup level record in `/etc/budate` (-u). Only the root file system is backed up, not mounted file systems.

2. To back up all files modified since the last level 0 backup:

```
backup -1 -u /
```

3. To back up selected files:

```
find $HOME -print | backup -v -i
```

This backs up all of the user's files, displaying a progress report as each file is copied (-v). The -i flag causes **backup** to read from standard input the names of files to be backed up. In this example, the **find** command supplies the list of file names. For more information about this command, see "**find**" on page 326.

4. To back up an entire minidisk:

```
backup -mf/dev/rmt0 /xyz
```

This backs up the entire minidisk that contains the file system `xyz`, copying it to the streaming tape (`/dev/rmt0`).

## Files

<code>/etc/filesystems</code>	Read for default parameters.
<code>/etc/budate</code>	Log for most recent backup dates.
<code>/dev/rfd0</code>	Default backup device.
<code>/dev/rhd0</code>	Default file system.

## Related Information

The following commands: “**find**” on page 326, “**format**” on page 331, and “**restore**” on page 596.

The **backup** and **filesystems** files and the **tape** special file in *AIX Operating System Technical Reference*.

“Backing up Files and File Systems” in *Managing the AIX Operating System*.

## banner

---

## banner

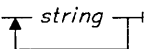
---

### Purpose

Writes character strings in large letters to standard output.

### Syntax

```
banner string
```



OL805080

### Description

The **banner** command writes character *strings* to standard output in large letters. Each line in the output can be up to 10 uppercase or lowercase characters long. On output, all characters appear in uppercase, with the lowercase input characters appearing smaller than the uppercase input characters.

### Examples

1. To display a banner at the work station:  

```
banner SMILE!
```
2. To display more than one word on a line, enclose the text in quotation marks:  

```
banner "Out to" Lunch
```

This displays Out to on one line, and Lunch on the next.
3. To print a banner:  

```
banner We like Computers | print
```

### Related Information

The following command: “**echo**” on page 278.

---

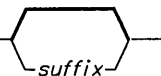
## basename, dirname

---

### Purpose

Returns the base name of a string parameter.

### Syntax

basename — *string* —  —

The diagram shows a horizontal line representing a string. A bracket is drawn below the line, starting from a point and extending to the right, with the word "suffix" written inside the bracket. This indicates that the portion of the string after the bracket is the suffix to be removed.

OL805085

dirname — *path* —

OL805047

### Description

The **basename** command reads the *string* specified on the command line, deletes any prefix that ends with a / (slash), as well as any specified *suffix*, if it is present, and writes the remaining base file name to standard output.

**Note:** A **basename** of / is null and is considered an error.

The **dirname** command writes to standard output all but the last part of the specified *path* name (all but the part following the last /).

The **basename** and **dirname** commands are generally used inside **command substitutions** within a shell procedure to specify an output file name that is some variation of a specified input file name. For more information, see “Command Substitution” on page 647.

### Examples

1. To display the base name of a shell variable:

```
basename $WORKFILE
```

This displays the base name of the value assigned to the shell variable WORKFILE. If WORKFILE is set to /u/jim/program.c, then program.c is displayed.

## basename

---

2. To construct a file name that is the same as another file name, except for its suffix:

```
OFILE=`basename $1 .c`.o
```

This assigns to OFILE the value of the first positional parameter (\$1), but with its .c suffix changed to .o. If \$1 is /u/jim/program.c, then OFILE becomes program.o. Because program.o is only a base file name, it identifies a file in the current directory.

The `` (grave accents) perform command substitution.

3. To construct the name of a file located in the same directory as another:

```
AOUTFILE=`dirname $TEXTFILE`/a.out
```

This sets the shell variable AOUTFILE to the name of an **a.out** file that is in the same directory as TEXTFILE. If TEXTFILE is /u/fran/prog.c, then the value of dirname \$TEXTFILE is /u/fran and AOUTFILE becomes /u/fran/a.out.

## Related Information

The following command: “sh” on page 637.

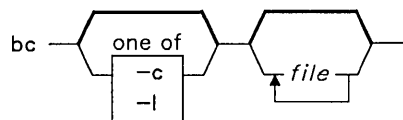
---

**bc**


---

**Purpose**

Provides an interpreter for arbitrary-precision arithmetic language.

**Syntax**

OL805081

**Description**

The **bc** command is an interactive process that provides unlimited precision arithmetic. It is a preprocessor for the **dc** command. **bc** invokes **dc** automatically, unless the **-c** (compile only) flag is specified. If the **-c** flag is specified, the output from **bc** goes to the standard output.

The **bc** command lets you specify an input and output base in decimal, octal, or hexadecimal (the default is decimal). The command also has a scaling provision for decimal point notation. The syntax for **bc** is similar to that of the C language.

The **bc** command takes input first from the specified *file*. When **bc** reaches the end of the input *file*, it reads standard input.

The following description of syntax for **bc** uses the following abbreviations: *L* means letters a-z; *E* means expressions; *S* means statements.

**Names**

Simple variables: *L*

Array elements: *L[E]*

The words **ibase**, **obase**, and **scale**.

Comments are enclosed in `/*` and `*/`.

## Other Operands

Arbitrarily long numbers with optional sign and decimal point.

( *E* )

sqrt ( *E* )

length ( *E* )      number of significant decimal digits

scale ( *E* )      number of digits to the right of the decimal point

L ( *E*, . . . , *E* )

## Operators

+ - \* / % ^ (% is remainder; ^ is power)

+ + -- (prefix and postfix; apply to names)

= = < = > = != < >

= = + =. =\* =/ =% =^

## Statements

*E*

{ *S*; . . . ; *S* }

if ( *E* ) *S*

while ( *E* ) *S*

for ( *E*; *E*; *E* ) *S*

(null statement)

break

quit

## Function Definitions

define L ( *L*, . . . , *L* ) {

    auto *L*, . . . , *L*

*S*; . . . *S*

    return ( *E* )

}

## Functions in -l Math Library

**s(x)** sine  
**c(x)** cosine  
**e(x)** exponential  
**l(x)** log  
**a(x)** arctangent  
**j(n,x)** Bessel function

All function parameters are passed by value.

The value of a statement that is an expression is displayed unless the main operator is an assignment. A semicolon or new-line character separates statements. Assignments to **scale** controls the number of decimal places printed on output and maintained during multiplication, division, and exponentiation. Assignments to **ibase** or **obase** set the input and output number radix respectively.

The same letter may refer to an array, a function, and a simple variable simultaneously. All variables are global to the program. "Auto" variables are pushed down during function calls. When you use arrays as function parameters, or define them as automatic variables, empty square brackets must follow the array name.

All **for** statements must have all three E's.

The **quit** statement is interpreted when read, not when executed.

## Flags

- c Compiles *file*, but does not invoke **dc**.
- l Includes a library of math functions.

## Examples

1. To use **bc** as a calculator:

```

You: bc
      1/4
System: 0
You: scale = 1 /* Keep 1 decimal place */
      1/4
System: 0.2
You: scale = 3 /* Keep 3 decimal places */
      1/4
System: 0.250
You: 16+63/5
System: 28.600
  
```



```
You: (16+63)/5
System: 15.800
You: 71/6
System: 11.833
You: 1/6
System: 0.166
```

You may type the comments (enclosed in `/* */`), but they are provided only for your information. **bc** displays the value of each expression when you press the **Enter** key, except for assignments.

When you enter **bc** expressions directly from the keyboard, press **END OF FILE (Ctrl-D)** to end the **bc** session and return to the shell command line.

2. To convert numbers from one base to another:

```
You: bc
      obase = 16      /* Display numbers in Hexadecimal */
      ibase = 8       /* Input numbers in Octal          */
      12
System: A
You: 123
System: 53
You: 123456
System: A72E
```

When you enter **bc** expressions directly from the keyboard, press **END OF FILE (Ctrl-D)** to end the **bc** session and return to the shell command line.

3. To write and run C-like programs:

```
You: bc -l prog.bc
      e(2)      /* e squared */
System: 7.38905609893065022723
You: f(5)      /* 5 factorial */
System: 120
You: f(10)     /* 10 factorial */
System: 3628800
```

This interprets the **bc** program saved in `prog.bc`, then reads more **bc** statements from the work station keyboard. Starting **bc** with the `-l` flag makes the math library available. This example uses the `e` (exponential) function from the math library, and `f` is defined in the program file `prog.bc` as:

```

/* compute the factorial of n */
define f(n) {
    auto i, r;

    r = 1;
    for (i=2; i<=n; i++) r =* i;
    return (r);
}

```

**Note:** The statement following a **for** or **while** statement must begin on the same line.

When you enter **bc** expressions directly from the keyboard, press END OF FILE (**Ctrl-D**) to end the **bc** session and return to the shell command line.

4. To convert an infix expression to reverse polish notation (RPN):

```

You: bc -c
      (a * b) % (3 + 4 * c)
System: 1a1b* 3 4|c*+%ps.

```

This compiles the **bc** infix-notation expression into one that the **dc** command can interpret. **dc** evaluates extended RPN expressions.

In the compiled output, the **l** (ell) before each variable name is the **dc** subcommand to load the value of the variable onto the stack. The **p** displays the value on top of the stack, and the **s.** discards the top value by storing it in register **.** (dot).

You can save the RPN expression in a file for **dc** to evaluate later by redirecting the standard output of this command. For more details, see “Redirection of Input and Output” on page 649.

When you enter **bc** expressions directly from the keyboard, press END OF FILE (**Ctrl-D**) to end the **bc** session and return to the shell command line.

## Files

/usr/lib/lib.b	Mathematical library.
/usr/bin/dc	Desk calculator proper.

## Related Information

The following command: “**dc**” on page 222.

# bdiff

---

## bdiff

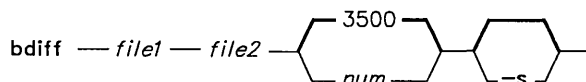
---

### Purpose

Uses **diff** to find differences in very large files.

### Syntax

```
bdiff — file1 — file2 — 3500 — num — -s —
```



OL805083

### Description

The **bdiff** command compares *file1* and *file2* and writes information about their differing lines to standard output. If either file name is - (minus), **bdiff** reads standard input. The **bdiff** command is used like **diff** to find lines that must be changed in two files to make them identical (see “**diff**” on page 246). Its primary purpose is to permit processing of files that are too large for **diff**.

The **bdiff** command ignores lines common to the beginning of both files, splits the remainder of each file into *num*-line segments, and calls **diff** to compare the corresponding segments. In some cases, the 3500 line default for *num* is too large for **diff**. If **diff** fails, specify a smaller value for *num* and try again.

The output of **bdiff** has the same format as that of **diff**. **bdiff** adjusts line numbers to account for the segmenting of the files. Note that because of the file segmenting, **bdiff** does not necessarily find the smallest possible set of file differences.

### Flag

**-s** Suppresses error messages from **bdiff**. (Note that the **-s** flag does not suppress error messages from **diff**).

### Example

To display the differences between *chap1* and *chap1.bak*:

```
bdiff chap1 chap1.bak
```

## Files

/tmp/bd\*      Temporary files.

## Related Information

The following command: “**diff**” on page 246.

## bfs

---

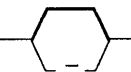
## bfs

---

### Purpose

Scans files.

### Syntax

bfs  file

OL805084

### Description

The **bfs** command reads a *file* but does not do any processing of it, allowing you to scan but not edit it.

The **bfs** command is basically a read-only version of the **ed** command, except it can process much larger files and it has some additional subcommands. Input files can be up to 32K lines long, with up to 255 characters per line. **bfs** is usually more efficient than **ed** for scanning a file, because the file is not copied to a buffer. It is most useful for identifying sections of a large file where you can use the **csplit** command to divide it into more manageable pieces for editing.

If you enter the **P** subcommand, **bfs** prompts you with an \* (asterisk). You can turn off prompting by entering a second **P**. **bfs** displays error messages when prompting is turned on.

### Forward and Backward Searches

The **bfs** command supports all the address expressions described under “**ed**” on page 280. In addition, you can instruct **bfs** to search forward or backward through the file, with or without wrap-around. If you specify a forward search with wrap-around, **bfs** continues searching from the beginning of the file after it reaches the end of the file. If you specify a backward search with wrap-around, it continues searching backwards from the end of the file after it reaches the beginning. The symbols for specifying the four types of search are as follows:

- /pattern/* Searches forward with wrap-around for the pattern.
- ?pattern?* Searches backward with wrap-around for the pattern.
- >pattern>* Searches forward without wrap-around for the pattern.
- <pattern<* Searches backward without wrap-around for the pattern.

The pattern matching routine of **bfs** differs somewhat from the one used by **ed** and includes additional features (see the **regcmp** subroutine in *AIX Operating System Technical Reference*). There is also a slight difference in **mark names**: only lowercase letters **a** through **z** may be used, and all 26 marks are remembered.

## Flags

- Suppresses the display of file sizes. Normally, **bfs** displays the size in bytes of the file being scanned.

## Subcommands

The **e**, **g**, **v**, **k**, **n**, **p**, **q**, **w**, **=**, **!** and null subcommands operate as explained under “**ed**” on page 280. Subcommands such as **-**, **+++**, **+++ =**, **-12**, and **+4p** are accepted. Note that **1,10p** and **1,10** both display the first ten lines. The **f** subcommand displays only the name of the file being scanned; there are no remembered file names. The **w** subcommand is independent of output diversion, truncation, or compression (see the **xo**, **xt**, and **xc** subcommands on page 91). **Compressed output** has strings of tabs and blanks reduced to one blank and blank lines suppressed.

The following additional subcommands are available:

- |  |  |
|--|--|
| <b>xf</b> <i>file</i>                  | Reads <b>bfs</b> subcommands from the <i>file</i> . When <b>bfs</b> reaches the end of file or receives an <b>INTERRUPT</b> signal or if an error occurs, <b>bfs</b> resumes scanning the file that contains the <b>xf</b> subcommand. These <b>xf</b> subcommands may be nested to a depth of 10.   |
| <b>xo</b> [ <i>file</i> ]              | Sends further output from the <b>p</b> and null subcommands to the named <i>file</i> , which is created with read and write permission granted to all users. If you do not specify a <i>file</i> parameter, <b>bfs</b> writes to standard output. Note that each redirection to a file creates the specified file, deleting an existing file if necessary.   |
| <b>:label</b>                          | Positions a <i>label</i> in a subcommand file. The <i>label</i> is ended with a new-line character. Blanks between the <b>:</b> (colon) and the start of the <i>label</i> are ignored. This subcommand may be used to insert comments into a subcommand file, since labels need not be referenced.   |
| <b>[addr1[,addr2]]xb/pattern/label</b> | Sets the current line to the line containing <i>pattern</i> and jumps to <i>label</i> in the current command file if <i>pattern</i> is matched within the designated range of lines. The jump fails under any of the following conditions: <ul style="list-style-type: none"> <li>• Either <i>addr1</i> or <i>addr2</i> is not between the first and last lines of the file.</li> <li>• <i>addr2</i> is less than <i>addr1</i>.</li> <li>• The pattern does not match at least one line in the specified range, including the first and last lines.</li> </ul> |

This subcommand is the only one that does not issue an error message on bad addresses, so it may be used to test whether addresses are bad before other subcommands are run. Note that the subcommand:

```
xb/^/label
```

is an *unconditional jump*.

The **xb** subcommand is allowed only if it is read from some place other than a work station. If it is read from a pipe, only a *downward jump* is possible.

**xt** *number* Truncates output from the **p** and null subcommands to *number* characters. The default *number* is 255.

**xv**[*digit*] [*value*] Assigns the specified *value* to the variable named *digit* (0 through 9). You can put one or more spaces between *digit* and *value*. For example:

```
xv5 100
xv6 1,100p
```

assigns the value 100 to the variable 5 and the value 1,100p to the variable 6.

To reference a variable, put a % (percent sign) in front of the variable name. Given the preceding assignments for variables 5 and 6, the following three subcommands:

```
1,%5p
1,%5
%6
```

each display the first 100 lines of a file. To escape the special meaning of %, precede it with a \ (backslash). For example:

```
g/".*\[c]ds]/p
```

matches and lists lines containing **printf** variables (%c, %d, or %s).

You can also use the **xv** subcommand to assign the first line of command output as the *value* of a variable. To do this, make the first character of *value* an ! (exclamation point), followed by the command name. For example:

```
xv5 !cat junk
```

stores the first line of the file junk in the variable 5. To escape the special meaning of ! as the first character of *value*, precede it with a \ (backslash). For example:

```
xv7 \!date
```

	stores the value <code>!date</code> in the variable <code>?</code> .
<b>xbz</b> <i>label</i>	Tests the last saved exit value from a shell command and jumps to <i>label</i> in the current command file if the value is zero.
<b>xbn</b> <i>label</i>	Tests the last saved exit value from a shell command and jumps to <i>label</i> in the current command file if the value is not zero.
<b>xc</b> [ <i>switch</i> ]	Turns compressed output mode on or off. (Compressed output mode suppresses blank lines and replaces multiple blanks and tabs with a single space.)  If <i>switch</i> is 1, output from the <b>p</b> and null subcommands is compressed; if <i>switch</i> is 0 it is not. If you do not specify <i>switch</i> , the current value of <i>switch</i> reverses. Initially, <i>switch</i> is set to 0.

## Related Information

The following commands: “**csplit**” on page 202 and “**ed**” on page 280.

The **regcmp** subroutine in *AIX Operating System Technical Reference*.



**bj**

---

**bj**

---

## Purpose

Plays blackjack.

## Syntax

`/usr/games/bj`—

OL805187

## Description

The **bj** game plays the the role of the dealer in blackjack. The following rules apply.

The bet is \$2 every hand. If you draw a *natural* (blackjack), you win \$3. If the dealer draws a natural, you lose \$2. If you and the dealer both have naturals, you exchange no money (a *push*). If the dealer has an ace showing, you can make an *insurance* bet on the chance that the dealer has a natural, winning \$2 if the dealer has a natural and lose \$1 if not. If you are dealt two cards of the same value, you can *double*, that is, play two hands, each of which begins with one of these cards, betting \$2 on each hand. If the value of your original hand is 10 or 11, you can *double down*, that is, double the bet to \$4 and receive exactly one more card in that hand.

Under normal play, you can draw a card (*hit*) as long as your cards total 21 or less. If the cards total more than 21, you *bust* and the dealer wins the bet. When you *stand* (decide not to hit), the dealer hits until he has a total of 17 or more. If the dealer busts, you win. If both you and the dealer stand, the one with the higher total wins. A tie is a push.

The **bj** command deals, keeps score, and asks the following questions at appropriate times: ? (Do you want a hit?) Insurance? Double? Double down?. To answer “yes,” press **y**; to answer “no,” press the **Enter** key.

The dealer tells you whenever the deck is being shuffled and displays the *action* (total bet) and *standing* (total won or lost). To quit the game, press INTERRUPT (**Alt-Pause**); **bj** displays the final action and standing and exits.

---

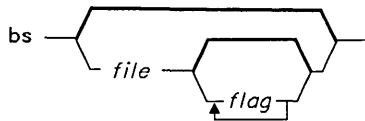
# bs

---

## Purpose

Compiles and interprets modest-sized programs.

## Syntax



OL805167

## Description

This compiler/interpreter provides interactive program development and debugging. To simplify program testing, it minimizes formal data declaration and file manipulation, allows line-at-a-time debugging, and provides trace and dump facilities and run-time error messages.

The optional command line parameter *file* specifies a file of program statements that the compiler reads before it reads from the standard input. By default, statements read from this file are compiled for later execution. Likewise, statements entered from the standard input are normally executed immediately (see the **compile** keyword on page 96 and the **execute** keyword on page 96). Unless the final operation is assignment, the result of an immediate expression statement is displayed.

Additional command line *flags* can be passed to the program using the built-in functions **arg** and **narg** (explained in more detail on page 100).

Program lines must conform to one of the following formats:

```
statement  
label statement
```

The interpreter accepts labeled statements only when it is compiling statements. A *label* is a *name* immediately followed by a colon. A label and a variable can have the same name. If the last character of a line is a \ (backslash), the statement continues on the following physical line.

A statement consists of either an expression or a keyword followed by zero or more expressions.

## Statement Syntax

<b>break</b>	Exits the innermost <b>for</b> or <b>while</b> loop.
<b>clear</b>	Clears the symbol table and removes compiled statements from memory. A <b>clear</b> is always executed immediately.
<b>compile</b> [ <i>expr</i> ]	Causes succeeding statements to be compiled (overrides the immediate execution default). The optional expression is evaluated and used as a file name for further input. In this latter case, the symbol table and memory are cleared first. <b>compile</b> is always executed immediately.
<b>continue</b>	Transfers control to the loop-continuation test of the current <b>for</b> or <b>while</b> loop.
<b>dump</b> [ <i>name</i> ]	Displays the name and current value of every global variable or, optionally, of the named variable. After an error or interrupt, <b>dump</b> displays the number of the last statement and (possibly) the user-function trace.
<b>exit</b> [ <i>expr</i> ]	Returns to the system level. The expression is returned as process status.
<b>execute</b>	Changes to immediate execution mode (pressing INTERRUPT [Alt-Pause] has the same effect). This statement does not cause stored statements to execute (see <b>run</b> on page 98).
<b>for</b> <i>name = expr expr statement</i>	
<b>for</b> <i>name = expr expr statement . . .</i>	
<b>next</b>	
<b>for</b> <i>expr, expr, expr statement</i>	
<b>for</b> <i>expr, expr, expr statement . . .</i>	
<b>next</b>	Repetitively performs, under the control of a named variable, a statement (first format) or a group of statements (second format). The variable takes on the value of the first expression, then is increased by one on each loop until it exceeds the value of the second expression. The third and fourth formats require three expressions separated by commas. The first of these is the initialization, the second is the test (true to continue), and the third is the loop-continuation action.
<b>fun</b> <i>f</i> ( <i>[a, . . . ]</i> ) [ <i>v, . . . ]</i> <i>statement . . .</i>	
<b>nuf</b>	Defines the function name ( <i>f</i> ), parameters ( <i>a</i> ), and local variables ( <i>v</i> ) for a user-written function. Up to 10 parameters and local variables are allowed. Such names cannot be arrays, nor can they be I/O associated. Function definitions may not be nested.

---

<b>freturn</b>	Signals the failure of a user-written function. Without interrogation, <b>freturn</b> returns zero. (See the unary interrogation operator <b>?</b> discussed on page 99.) With interrogation, <b>freturn</b> transfers to the interrogated expression, possibly bypassing intermediate function returns.
<b>goto name</b>	Passes control to the compiled statement with the matching label.
<b>ibase n</b>	Sets the input base to <i>n</i> . The only supported values for <i>n</i> are 8, 10 (the default), and 16. Hexadecimal values 10-15 are entered as alphabetic characters a-f. A leading digit is required when a hexadecimal number begins with an alphabetic character (for example, f0a must be entered as 0f0a). <b>ibase</b> is always executed immediately.
<b>if expr statement</b>	
<b>if expr</b> <i>statement . . .</i> [ <b>else</b> <i>statement . . .</i> ] <b>fi</b>	Performs a statement (first format) or group of statements (second format) if the expression evaluates to nonzero. The strings 0 and "" (null) evaluate as zero. In the second format, an optional <b>else</b> allows a group of statements to be performed when the first group is not. The only statement permitted on the same line with an <b>else</b> is an <b>if</b> ; only other <b>fis</b> can be on the same line with a <b>fi</b> . You can combine <b>else</b> and <b>if</b> into <b>elif</b> . Only a single <b>fi</b> is required to close an <b>if . . . elif . . . [else . . . ]</b> sequence.
<b>include expr</b>	The expression must evaluate to the name of a file containing program statements. Such statements become part of the program being compiled. <b>include</b> statements may not be nested, and are always executed immediately.
<b>obase n</b>	Sets the output base to <i>n</i> . The only supported values for <i>n</i> are 8, 10 (the default), and 16. Hexadecimal values 10-15 are entered as alphabetic characters a-f. A leading digit is required when a hexadecimal number begins with an alphabetic character (that is, f0a must be entered as 0f0a). Like <b>ibase</b> , <b>obase</b> is always executed immediately.
<b>onintr label</b> <b>onintr</b>	Provides program control of interrupts. In the first format, control passes to the label given, just as if a <b>goto</b> had been performed when <b>onintr</b> was executed. The effect of the <b>onintr</b> statement is cleared after each interrupt. In the second format, pressing INTERRUPT (Alt-Pause) ends <b>bs</b> .
<b>return [expr]</b>	Evaluates the expression and passes the result back as the value of a function call. If you do not provide an expression, the function returns zero.

<b>run</b>	Passes control to the first compiled statement. The random number generator is reset. If a file contains a <b>run</b> statement, it should be the last statement; <b>run</b> is always executed immediately.
<b>stop</b>	Stops execution of compiled statements and returns to immediate mode.
<b>trace</b> [ <i>expr</i> ]	Controls function tracing. If you do not provide an expression or if it evaluates to zero, tracing is turned off. Otherwise, a record of user-function calls/returns will be written. Each <b>return</b> decreases by one the <b>trace</b> expression value.
<b>while</b> <i>expr statement</i>	
<b>while</b> <i>expr</i> <i>statement . . .</i>	
<b>next</b>	<b>while</b> is similar to <b>for</b> except that only the conditional expression for loop continuation is given.
<b>! AIXcmd</b>	Runs an AIX command, then returns control to <b>bs</b> .
<b>#comment</b>	Inserts a comment line.

## Expression Syntax

<i>name</i>	Specifies a variable or, when followed immediately by a colon, a label. Names are composed of a letter (uppercase or lowercase) optionally followed by letters and digits. Only the first six characters of a name are significant. Except for names declared locally in <b>fun</b> statements, all names are global. Names can take on numeric (double float) values or string values or be associated with input/output (see the built-in function <b>open</b> on page 102).
<i>name</i> ( <i>[expr</i> [, <i>expr</i> ] . . . )	Calls function <i>name</i> and passes to it the parameters in parentheses. Except for built-in functions (listed in the following text), <i>name</i> must be defined in a <b>fun</b> statement. Function parameters are passed by value.
<i>name</i> [ <i>expr</i> [, <i>expr</i> ] . . . ]	References either arrays or tables (see built-in function <b>table</b> on page 103). For arrays, each expression is truncated to an integer and used as a specifier for the name. The resulting array reference is syntactically identical to a name; a [1,2] is the same as a [1] [2]. The truncated expressions must be values between 0 and 32767.
<i>number</i>	Represents a constant numerical value. This number can be expressed in integer, decimal, or scientific notation (it can contain digits, an optional decimal point, and an optional <i>e</i> followed by a possibly signed exponent).
<i>string</i>	Character string delimited by " " (double quotation marks). The \ (backslash) is an escape character that allows the double quotation mark ("), new-line character (\n), carriage return (\r), backspace (\b), and tab

(\t) characters to appear in a string. When not immediately followed by these special characters, \ stands for itself.

*(expr)* Parentheses alter the normal order of evaluation.

*(expr, expr[, expr] . . . ) [expr]*

The bracketed expression outside the parentheses functions as a subscript to the list of expressions within the parentheses. List elements are numbered from the left, starting at zero. The expression:

(False, True) [ a == b ]

has the value **True** if the comparison is true.

*expr op expr* Except for the assignment, concatenation, and relational operators, both operands are converted to numeric form before the operator is applied.

## Unary Operators

*?expr* The interrogation operator (?) tests for the success of the expression rather than its value. It is useful for testing end of file, for testing the result of the **eval** built-in function, and for checking the return from user-written functions (see **freturn** on page 97). An interrogation **trap** (end of file, for example), causes an immediate transfer to the most recent interrogation, possibly skipping assignment statements or intervening function levels.

*-expr* Negates the expression.

*++name* Increases by one the value of the variable (or array reference).

*--name* Decreases by one the value of the variable.

*!expr* The logical negation of the expression.

## Binary Operators (in increasing precedence)

**=** The assignment operator. The left operand must be a name or an array element. It acquires the value of the right operand. Assignment binds right to left; all other operators bind left to right.

**\_** The concatenation operator (the underline character).

**& |** logical AND, logical OR. The result of

*expr & expr*

is 1 (true) only if both of its parameters are nonzero (true); it is 0 (false) if one or both of its parameters are 0 (false). The result of

*expr | expr*

is 1 (true) if one or both of its expressions are nonzero (true); it is 0 (false) only if both of its expressions are 0 (false). Both operators treat a null string as a zero.

< <= > >= == !=

The relational operators (< less than, <= less than or equal to, > greater than, >= greater than or equal to, == equal to, != not equal to) return 1 if the specified relation is True. They return 0 (false) otherwise. Relational operators at the same level extend as follows: **a > b > c** is the same as **a > b & b > c**. A string comparison is made if both operands are strings. The comparison is based on the collating sequence specified in the environment variable **NLCTAB**.

+ -

Addition and subtraction.

\* / %

Multiplication, division, and remainder.

^

Exponentiation.

## Functions Dealing With Arguments

**arg(*i*)** Returns the value of the *i*-th actual argument at the current function call level. At level zero, **arg** returns the *i*-th command-line argument. For example, **arg(0)** returns **bs**.

**narg( )** Returns the number of arguments passed. At level zero, it returns the command line argument count.

## Mathematical Functions

**abs(*x*)** Returns the absolute value of *x*.

**atan(*x*)** Returns the arctangent of *x*.

**ceil(*x*)** Returns the smallest integer not less than *x*.

**cos(*x*)** Returns the cosine of *x*.

**exp(*x*)** Returns e raised to the power *x*.

**floor(*x*)** Returns the largest integer not greater than *x*.

**log(*x*)** Returns the natural logarithm of *x*.

**rand( )** Returns a uniformly distributed random number between zero and one.

**sin(*x*)** Returns the sine of *x*.

**sqrt(*x*)** Returns the square root of *x*.

## String Functions

- size(*s*)** Returns the size (length in bytes) of *s*.
- format(*f*, *a*)** Returns the formatted value of *a*, *f* being a format specification string in the style of the **printf** subroutine. Use only the **%...f**, **%...e**, and **%...s** formats.
- index(*x*, *y*)** Returns a number that is the first position in *x* containing a character that any of the characters in *y* matches. If there is no match, **index** yields zero. For two-byte extended characters, the index functions returns the location of the first byte.
- trans(*s*, *f*, *t*)** Translates characters in the source string *s* which match characters in *f* into characters having the same position in *t*. Source characters that do not appear in *f* are copied unchanged into the translated string. If string *f* is longer than *t*, source characters that match characters found in the excess portion of *f* do not appear in the translated string.
- subst(*s*, *start*, *length*)**  
Returns the substring of *s* defined by *starting* position and *length*.
- match(*string*, *pattern*)**  
**mstring(*n*)** This function returns the number of characters in *string* that match *pattern*. The characters **.**, **\***, **?**, **[**, **]**, **^** (when inside square brackets), **\(** and **\)** have the following special meanings (see “**ed**” on page 280 for a more detailed discussion of this special notation):
- .** Matches any character except the new-line character.
  - \*** Matches zero or more occurrences of the pattern element that it follows (for example, **.\*** matches zero or more occurrences of any character except the new-line character).
  - \$** Specifies the end of the line.
  - [.-]**
  - [...]** Matches any one character in the specified range (**[.-]**) or list (**[...]**), including the first and last characters.
  - [^.-]**
  - [^...]** Matches any character except the new-line character and the remaining characters in the range or list. A circumflex (**^**) has this special meaning only when it immediately follows the left bracket.
  - [].-]**
  - []...]** Matches **]** or any character in the list. The right square bracket does not terminate such a list when it is the first character within it (after an initial **^**, if any).
  - \(...\)** Marks a substring and matches it exactly.



To succeed, a pattern must match from the beginning of the string. It also matches the longest possible string. Consider, for example:

```
match('a123ab123', ".*\[a-z]\") == 6
```

In this instance, `.*` matches `a123a` (the longest string that precedes a character in the range `a-z`); `\([a-z]\)` matches `b`, giving a total of six characters matched in the string. In an expression such as `[a-z]`, the minus means “through” according to the current collating sequence. A collating sequence may define *equivalence classes* for use in character ranges. See the “Overview of International Character Support” in *Managing the AIX Operating System* for more information on collating sequences and equivalence classes.

The `mstring` function returns the *n*th substring in the last call to `match` (*n* must be between 1 and 10 inclusive).

## File-Handling Functions

`open(name, file, mode)`

`close(name)`

The *name* parameter must be a legal variable name (passed as a string). For `open`, the *file* parameter may be:

- A `0`, `1`, or `2` for standard input, output, or error output, respectively
- A string representing a file name
- A string beginning with an `!`, representing a command to be run (via `sh -c`).

The *mode* flag must be either `r` (read), `w` (write), `W` (write without new-line character), or `a` (append). After a `close`, the *name* becomes an ordinary variable. The initial associations are:

```
open("get", 0, "r")
open("put", 1, "w")
open("puterr", 2, "w")
```

`access(p, m)`

Performs the `access` system call. Parameter *p* is the path name of a file; *m* is a bit pattern representing the requested mode of access. This function returns a `0` if the request is permitted, `-1` if it is denied. (See *AIX Operating System Technical Reference* for a more extensive discussion of this system call.)

`ftype(s)`

Returns a single character indicating file type: `f` for regular file, `p` for FIFO (named pipe), `d` for directory, `b` for block special, or `c` for character special.

## Table Functions

**table**(*name*, *size*) A table in **bs** is an associatively accessed, one-dimensional array. "Subscripts" (called keys) are strings (numbers are converted). The *name* parameter must be a **bs** variable name (passed as a string). The *size* parameter sets the minimum number of elements to be allocated. On table overflow, **bs** writes an error message.

**item**(*name*, *i*)  
**key**( )

The **item** function accesses table elements sequentially (in normal use, there is an orderly progression of key values). Where the **item** function accesses values, the **key** function accesses the "subscript" of the previous **item** call. The *name* parameter should not be quoted. Since exact table sizes are not defined, the interrogation operator should be used to detect end-of-table; for example:

```
table("t",100)
.
.
.
#If word contains "party", the following expression
#adds one to the count of that word:
++t[word]
.
.
.
# To display the key/value pairs:
for i=0,?(s=item(t, i)), ++i if key() put=key()_"": "_s
```

**iskey**(*name*, *word*)

Tests whether the key *word* exists in the table *name* and returns one for true, zero for false.

## Miscellaneous Functions

**eval**(*string*) The string parameter is evaluated as an expression. The function is handy for converting numeric strings to numbers. **eval** can also be used as a crude form of indirection, as in:

```
name = "xyz"
eval("++" _name)
```

which increments the variable *xyz*. In addition, **eval** preceded by the interrogation operator permits you to control **bs** error conditions. For example:

```
?eval("open(\"X\", \"XXX\", \"r\")")
```

returns the value zero if there is no file named "XXX" (instead of halting your program). The following performs a **goto** to the label *L*: (if it exists):

```
label="L:"  
if!(?eval("goto" _label))puterr="no label"
```

**plot(request, args)**

The **plot** function produces output on devices recognized by the **tplot** command. The *requests* are as follows:

**Call**

**Function**

**plot(0, term)**

Causes further **plot** output to be piped into **tplot** with a flag of **-Tterm**.

**plot(1)**

"Erases" the plotter.

**plot(2, string)**

Labels the current point with *string*.

**plot(3, x1, y1, x2, y2)**

Draws the line between (x1, y1) and (x2, y2).

**plot(4, x, y, r)**

Draws a circle with center (x, y) and radius *r*.

**plot(5, x1, y1, x2, y2, x3, y3)**

Draws an arc (counterclockwise) with center (x1, y1) and endpoints (x2, y2) and (x3, y3).

**plot(6)**

Not implemented.

**plot(7, x, y)**

Makes the current point at (x, y).

**plot(8, xy)**

Draws a line from the current point to (x, y).

**plot(9, x, y)**

Draws a point at (x, y).

**plot(10, string)**

Sets the line mode to *string*.

**plot(11, x1, y1, x2, y2)**

Makes (x1, y1) the lower left corner of the plotting area and (x2, y2) the upper right corner of the plotting area.

**plot(12, x1, y1, x2, y2)**

Causes subsequent x (y) coordinates to be multiplied by x1 (y1) and then added to x2 (y2) before they are plotted. The initial scaling is **plot(12, 1.0, 1.0, 0.0, 0.0)**.

Some requests do not apply to all plotters. All requests except zero and 12 are implemented by piping characters to **tplot**.

**last()**

In immediate mode, **last** returns the most recently computed value.

## Related Information

The following commands: “**ed**” on page 280, “**sh**” on page 637, and “**tplot**” on page 762.

The **access** system call, the **printf** subroutine, and the **plot** file in *AIX Operating System Technical Reference*.

“Overview of International Character Support” in *Managing the AIX Operating System*.

# cal

---

## cal

---

### Purpose

Displays a calendar.

### Syntax

cal *month* *year*

OL805168

### Description

The **cal** command writes to standard output a calendar for the specified year or month.

The *month* parameter names the month for which you want the calendar. It can be a number between 1 and 12 for January through December, respectively.

The *year* parameter names the year for which you want the calendar. Since **cal** can display a calendar for any year from 1 to 9999, enter the full *year* rather than just the last two digits.

### Examples

1. To display a calendar for February 1984 at your work station:

```
cal 2 1984
```

2. To print a calendar for 1984:

```
cal 1984 | print
```

3. To display a calendar for the year 84 A.D.:

```
cal 84
```

---

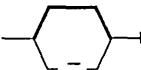
# calendar

---

## Purpose

Writes reminder messages to standard output.

## Syntax

calendar 

OL805169

## Description

The **calendar** command reads a file named **calendar**, which you create in your current (usually home) directory. It writes to standard output any line in the file that contains today's or tomorrow's date.

The **calendar** command recognizes date formats such as DEC. 7 or 12/7. It also recognizes the special character \* (asterisk). It interprets \*/7, for example, as signifying the seventh day of every month. **calendar** does not recognize formats such as 7 December, 7/12, or DEC. 7.

On Fridays, **calendar** writes all lines containing the dates for Friday, Saturday, Sunday, and Monday. It does not, however, recognize holidays, so "tomorrow" is the holiday rather than the next working day.

For you to get reminder service, your **calendar** should have read permission for others (see "**chmod**" on page 128).

## Flag

- Calls **calendar** for everyone having a file **calendar** in his home directory and sends any reminders by **mail**

## Example

To display information in the **calendar** file that pertains to the next two business days:

```
calendar
```

## calendar

---

A typical **calendar** file might look like this:

```
*/25 - Prepare monthly report
Aug. 12 - Fly to Denver
aug 23 - board meeting
Martha out of town - 8/23, 8/24, 8/25
8/24 - Mail car payment
sat aug/25 - beach trip
August 27 - Meet with Simmons
August 28 - Meet with Wilson
```

If today is Friday, August 24, then the **calendar** command displays:

```
*/25 - Prepare monthly report
Martha out of town - 8/23, 8/24, 8/25
8/24 - Mail car payment
sat aug/25 - beach trip
August 27 - Meet with Simmons
```

## Files

<code>\$HOME/calendar</code>	
<code>/usr/lib/calprog</code>	The program that determines dates.
<code>/etc/passwd</code>	Used to identify users.
<code>/tmp/cal*</code>	Temporary files.

## Related Information

The following commands: “**chmod**” on page 128 and “**mail**” on page 470.

---

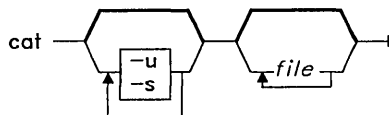
# cat

---

## Purpose

Concatenates or displays files.

## Syntax



OL805086

## Description

The `cat` command reads each *file* in sequence and writes it to standard output. If you do not specify *file* or specify - (minus) instead of a *file*, `cat` reads from standard input.

**Warning:** Do not redirect output to one of the input files using the `>` redirection symbol. If you do this, you will lose the original data in the input file because the shell truncates it before `cat` can read it (see “`sh`” on page 637).

## Flags

- s Does not display a message if `cat` cannot find an input file.
- u Does not buffer output.

## Examples

1. To display a file at the work station:

```
cat notes
```

This displays the data in the file `notes`. If the file is more than about 23 lines long, some of it will scroll off the screen. To list a file one page at a time, use the `pg` command. (See “`pg`” on page 553 for details.)

2. To concatenate several files:

```
cat section1.1 section1.2 section1.3 >section1
```



This creates a file named `section1` that is a copy of `section1.1` followed by `section1.2` and `section1.3`.

3. To suppress error messages about files that do not exist:

```
cat -s section2.1 section2.2 section2.3 >section2
```

If `section2.1` does not exist, this concatenates `section2.2` and `section2.3`. The result is the same if you do not use the `-s`, except that `cat` displays the error message:

```
cat: cannot open section2.1
```

You may want to suppress this message with the `-s` flag when you use the `cat` command in shell procedures.

4. To append one file to the end of another:

```
cat section1.4 >>section1
```

This appends a copy of `section1.4` to the end of `section1`.

**Note:** The `>>` appends data to the end of `section1`, but using `>` replaces the file. For more details, see “Redirection of Input and Output” on page 649.

5. To add text to the end of a file:

```
cat >>notes
Get milk on the way home
Ctrl-D
```

This adds the text `Get milk on the way home` to the end of `notes`. The `cat` command does not prompt, but waits silently for you to enter text. Pressing **Ctrl-D** indicates the end of the text to be added.

6. To concatenate several files with text entered from the keyboard:

```
cat section3.1 - section3.3 >section3
```

This concatenates `section3.1`, text from the keyboard, and `section3.3`.

7. To concatenate several files with output from another command:

```
li | cat section4.1 - >section4
```

This copies `section4.1`, followed by the output of the `li` command to a file named `section4`.

## Related Information

The following commands: “**cp**” on page 156, “**pr**” on page 561, and “**sh**” on page 637.

---

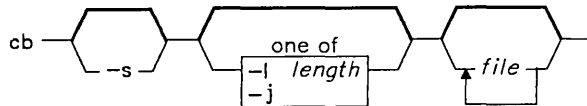
**cb**


---

## Purpose

Puts C source code into a form that is easily read.

## Syntax



OL805170

## Description

The **cb** command reads C programs from standard input or from specified *files* and writes them to standard output in a form that shows, through indentations and spacing, the structure of the code. When called without flags, **cb** does not split or join lines. Note that punctuation in preprocessor statements can cause indentation errors.

## Flags

- j Joins lines that are split.
- l *length* Splits lines that are longer than *length*.
- s Formats the source code according to the style of Kernighan and Ritchie in *The C Programming Language*. (Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1978.).

## Example

To create a version of `pgm.c` called `pgm.pretty.c` that is easy to read:

```
cb pgm.c > pgm.pretty.c
```

## Related Information

The following command: “**cc**” on page 112.

The discussion of **cb** in *AIX Operating System Programming Tools and Interfaces*.

**cc**

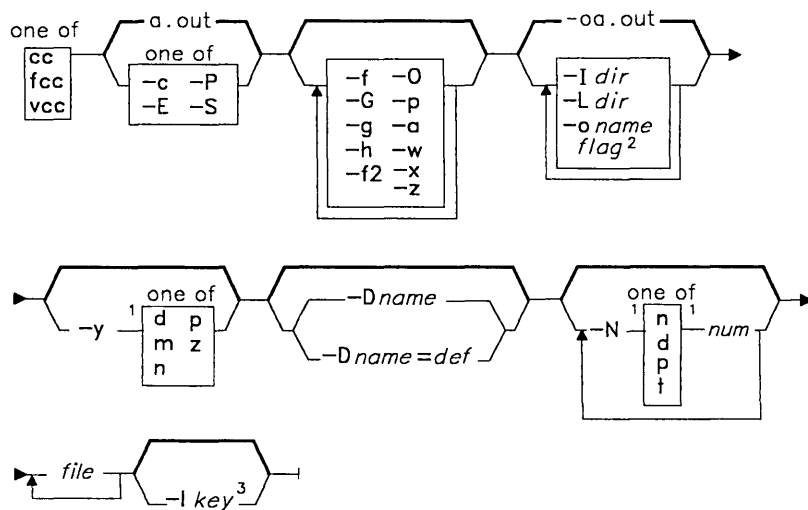
**cc**

## Purpose

Compiles C programs.

## Syntax

### Ordinary Operation



OL805171

<sup>1</sup> Do not put a blank between these items.

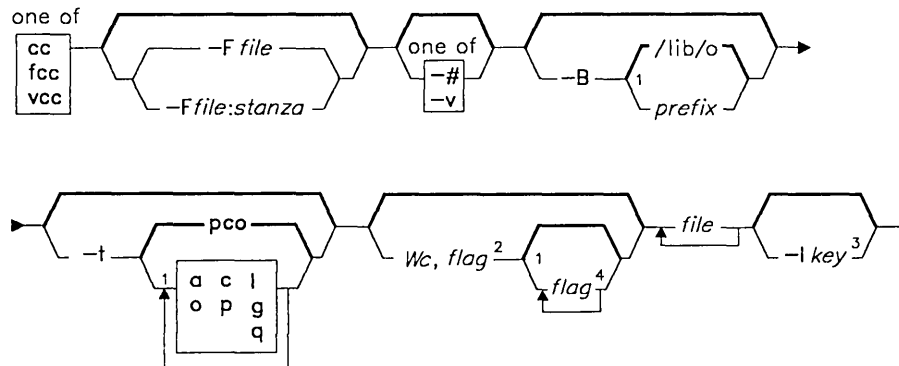
<sup>2</sup> Use any flag belonging to `as`, `cpp`, or `ld` (except `-lkey`).

<sup>3</sup> Put this flag last if used (see the `ld` command).

<sup>4</sup> Use any flag from the first diagram (except `-lkey`) or any flag belonging to `as`, `cpp` or `ld`.

OL805343

## Extended Functions and Debugging



OL805389

<sup>1</sup> Do not put a blank between these items.

<sup>2</sup> Use any flag belonging to `as`, `cpp`, or `ld` (except `-lkey`).

<sup>3</sup> Put this flag last if used (see the `ld` command).

<sup>4</sup> Use any flag from the first diagram (except `-lkey`) or any flag belonging to `as`, `cpp` or `ld`.

OL805343

## Description

The `cc` command runs the C compiler. It accepts files containing C source code, assembler source code, or object code and changes them into a form that the computer system can run. `cc` compiles and assembles source files and then links them with any specified object files, in the order listed on the command line. It puts the resulting executable program in a file named `a.out`.

The `fcc` command is a link to `cc` that compiles programs to run with the Floating-Point Accelerator. `fcc` should only be used on the ROM processor. It automatically uses the `-f` flag as well as special versions of the standard libraries that have been compiled for direct floating-point applications. Note that programs compiled with `fcc` can run only on systems that have installed the Floating-Point Accelerator.

The `vcc` command is a link to `cc` that compiles modules to be installed in the VRM. Use the `vrmfmt` command to convert the `a.out` file produced by the `vcc` command to a VRM-compatible object module. The syntax of this command is as follows:

```
vrmfmt infile [outfile]
```

The default output file name is `a.vrm`.

**cc**

---

The **cc** command runs the following programs. Each program processes the source file and then sends the results to the next program in the sequence:

**cpp**        The macro preprocessor.

**ccom0**     The first pass of the compiler.

**ccomq**     The intermediate code optimizer (if you specify the **-O** flag).

This program provides a variety of optimizations to the intermediate code, such as removing loop invariants, eliminating common subexpressions, and allocating registers. The following cannot be optimized:

- Functions that call **setjmp**
- Functions that contain **asm** statements

If you are compiling a large program and the flow optimizer runs out of space, the compiler stops the process and displays a message describing the problem.

**ccom1**     The second pass of the compiler.

**copt**      The optimizer (if you specify the **-O** flag).

**as**         The assembler.

**ld**         The linkage editor.

You can replace any or all of these passes with your own versions (see the **-B** and **-t** flags). Both **cc** and **fcc** use the **cc.cfg** configuration file, which specifies the standard runtime link options and libraries to be used with each version of the compiler.

## Input File Types

The **cc** command recognizes and accepts as input the following file types:

### *file.c*

The name of a C language source file should end with **.c**. After **cc** compiles this source file, it gives the resulting object file the same name, except that it ends in **.o** rather than **.c**. If you use one command both to compile and to load a single C program, the compiler normally deletes the **.o** file when it loads the program. If you use the **-c** flag, the compiler does not delete the **.o** file.

### *file.i*

The name of a file that contains preprocessed C source code ends in **.i**.

***file.o***

The name of an object file should end in **.o**. The **cc** command sends these files to the **ld** command.

***file.s***

The name of an assembly language source program should end with **.s**. After **cc** assembles this source file, it gives the resulting object file the same name, except that it ends in **.o** rather than **.s**.

**Flags**

The **cc** command recognizes several flags. In addition, flags intended to modify the action of the linkage editor (**ld**), the assembler (**as**), or the preprocessor (**cpp**) may also appear on the **cc** command line. **cc** sends any flags it does not recognize to these commands for processing. The following list includes the most commonly used **cpp** flags (**-D**, **-I**), and **ld** flags (**-l**, **-L**, **-o**). See “**as**” on page 64, “**cpp**” on page 163, and “**ld**” on page 427 for a complete list of additional flags.

**Note:** If you use the **-l** flag, it must be the last entry on the command line, following any file parameters.

**Ordinary Operation**

- |                     |   |
|---------------------|---|
| <b>-a</b>           | Reserves a register for extended addressing. You should use this flag if a compiled procedure creates a stack greater than 32,767 bytes. Because this flag causes the compiler to reserve a register for use by the assembler, it reduces the number of available registers by one. |
| <b>-c</b>           | Does not send the completed object file to the <b>ld</b> command. With this flag, the output of <b>cc</b> is a <b>.o</b> file for each <b>.c</b> or <b>.s</b> file.   |
| <b>-Dname[=def]</b> | Defines <i>name</i> as in a <b>#define</b> directive. The default <i>def</i> is 1.  |
| <b>-E</b>           | Runs the named C source file through only the preprocessor and writes the result to standard output.  |
| <b>-f</b>           | Generates code that uses the Floating-Point Accelerator. Programs compiled with this flag will run correctly only on AIX processors configured with the Floating-Point Accelerator.   |
| <b>-f2</b>          | Generates code that uses the Advanced Floating-Point Accelerator. Programs compiled with this flag will run correctly only on AIX processors configured with the Advanced Floating-Point Accelerator and an Advanced Processor Card.  |

- 
- g Produces additional information for use with the **sdb** command (the symbolic debugger).
  - G Indicates that global variables are volatile. The optimizer (**ccomq**) makes fewer transformations when you specify this flag. To make a particular variable volatile, add the “volatile” specification to its declaration.
  - h Treats files with the suffix **.h** in the same way as files with the suffix **.c**.
  - I*dir* Looks first in *dir*, then looks in the directories on the standard list for **#include** files with names that do not begin with / (slash).
  - l[*key*] Searches the specified library file, where *key* selects the file **libkey.a**. With no *key*, **-l** selects **libc.a**, the standard system library for C and assembly language programs. **ld** searches for this file in the directory specified by an **-L** flag, then in **/lib** and **/usr/lib**. The **ld** command searches library files in the order in which you list them on the command line.
  - L*dir* Looks in *dir* for files specified by **-l** keys. If it does not find the file in *dir*, **ld** searches the standard directories.
  - N[**ndpt**]*num* Changes the size of the symbol table (**n**), the dimension table (**d**), the constant pool (**p**), or the space for building the parse tree (**t**). Each table must be changed separately. The default size of the symbol table is 1500; the default size of the dimension table is 2000; the default size for the constant pool is 600; the default space for the parse tree is 1000.
  - o*name* Assigns *name* rather than **a.out** to the output file.
  - O Sends compiler output to the code optimizers.
  - p Prepares the program so that the **prof** command can generate an execution profile. The compiler produces code that counts the number of times each routine is called. If programs are sent to **ld**, the compiler replaces the startup routine with one that calls the **monitor** subroutine at the start (see *AIX Operating System Technical Reference* for a discussion of this subroutine), and writes a **mon.out** file when the program ends normally.
  - P Sends the specified C source file to the macro preprocessor and stores the output in a **.i** file.
  - Q! Turns off inlining. The following may be used:
    - ? Shows the reason for not inlining in the output file.
    - name,name* . . . Does not inline *name*
    - + *name,name* . . . Inlines *name*

- 
- | *num* Limits the size increase of the function into which inlining occurs to *num* intermediate operations. The default *num* is 100.
- | *#num* Limits the expansion of an individual call to *num* intermediate operators. The default *num* is 100.
- | *-@file* Reads a list of forbidden functions from *file*.
- | *+@file* Reads a list of requested functions from *file*.
- | Requesting a function to be inlined overrides size constraints.
- | **-S** Compiles the specified C programs, storing assembly language output in a *.s* file.
- | **-w** Prevents printing of warning messages about functions that cannot be optimized.
- | **-X** Produces an assembler listing. This is stored in a file that has the same name as the assembler source file, but with the extension *.lst* instead of *.s*.
- | **-y[dmnpz]** Specifies the rounding mode for floating-point constant folding. These modes are specified as follows:
- | **d** Disables floating-point constant folding.
- | **m** Rounds toward negative infinity.
- | **n** Rounds to nearest. This is the default action and applies to constant folding in all applicable passes of the compiler.
- | **p** Rounds toward positive infinity.
- | **z** Rounds toward 0.
- | **-z** Uses the **libm.a**, or one specified by the user, version of the following transcendental functions:
- | acos cos sin
- | asin exp sqrt
- | atan log tan
- | atan2 log10
- | If this flag is not used, the compiler generates calls to the AIX kernel, or the Advanced Floating Point Accelerator if possible. For more information, see **math.h** in *AIX Operating System Technical Reference*.



## Debugging

- Ffile[:stanza]** Uses an alternative *file* and/or *stanza* for **cc** configuration (see *AIX Operating System Technical Reference* for a discussion of the configuration file, **cc.cfg**). If used, this flag must be the first flag on the command line.
- v** Displays the trace as with **-#** and invokes the programs.
- #** Displays a trace of the actions to be taken (for example, invoking the preprocessor), without actually invoking any programs.

## Extended Functions

- Bprefix** Constructs path names for substitute preprocessor, compiler, optimizer, assembler, or linkage editor programs. *prefix* defines part of a path name to the new programs. To form the complete path name for each new program, **cc** adds *prefix* to the standard program names (see the discussion of the programs called by **cc** on page 114). For example, if you enter the command:

```
cc testfile.c -B/usr/jim/new
```

**cc** calls the following compiler programs:

1. /usr/jim/newcpp
2. /usr/jim/newccom0
3. /usr/jim/newccom1
4. /usr/jim/newas
5. /usr/jim/newld

Similarly, if you enter the command:

```
cc testfile.c -B/usr/jim/new/
```

**cc** calls the following compiler programs:

1. /usr/jim/new/cpp
2. /usr/jim/new/ccom
3. /usr/jim/new/ccom1
4. /usr/jim/new/as
5. /usr/jim/new/ld

The default *prefix* is **/lib/o**.

- t[pcqgoal]** Applies the **-B** flag instructions for constructing file names to only the designated preprocessor (**p**), compiler first (**c**), intermediate code optimizer (**q**), compiler second (**g**), optimizer (**o**), assembler (**a**), or linkage editor (**l**) passes. You can select any combination of **pcqgoal**.

The **-t** flag with no additional **p**, **c**, **q**, **g**, **o**, **a**, or **l** designates by default the preprocessor, compiler and optimizer programs (see the discussion of the programs called by **cc** on page 114).

If you do not specify the **-B** flag when you specify the **-t** flag, the default file name *prefix* is **/lib/n**.

**Note:** You can specify this *prefix* with the **-B** flag. However, depending on what combination of the **-B** and the **-t** flags you specify, *prefix* can have two possible default values. If you specify **-B** but no accompanying *prefix*, the default *prefix* is **/lib/o**. If you specify the **-t** flag without also specifying the **-B** flag, the default *prefix* is **/lib/n**.

**-Wc,flag1[,flag2 . . . ]**

Gives the listed flags to the compiler program *c*; *c* can be any one of the values [**pcqgoal**] discussed with the **-t** flag. For example, since both **ld** and **as** recognize a **-o** flag, use **-W** to specify the program to which the flag is to be sent. That is, **-Wl,-o** sends it to **ld**. **-Wa,-o** sends it to **as**.

## Examples

1. To compile and link a C program, creating an executable **a.out** file:

```
cc pgm.c
```

2. To compile a program, producing an object file to be linked later:

```
cc -c pgm.c
```

This compiles **pgm.c** and produces an object file named **pgm.o**.

3. To compile a program to run on the Floating-Point Accelerator:

```
fcc pgm.c
```

This compiles **pgm.c** using the special libraries **libfc.a** and **libfm.a** instead of the standard libraries **libc.a** and **libm.a**.

4. To view the output of the macro preprocessor:

```
cc -P -C pgm.c
```

This creates a file named **pgm.i** that contains the preprocessed program text including comments. To view this file, use an editor or see “**pg**” on page 553 **cc** passes the **-P** and **-C** flags to the preprocessor. See “**cpp**” on page 163 for more details about them.

5. To predefine macro identifiers:

```
cc -DBUFFERSIZE=512 -DDEBUG pgm.c
```

This assigns **BUFFERSIZE** the value 512 and **DEBUG** the value 1 before preprocessing. **cc** passes the **-D** flag to the preprocessor.

6. To use **#include** files located in nonstandard directories:

```
cc -I/u/jim/include pgm.c
```

This looks in the directory that contains `pgm.c` for the **#include** files with names enclosed in double quotes (" "), then in `/u/jim/include`, and then in the standard directories. It looks in `/u/jim/include` for **#include** file names enclosed in angle brackets (< >), then in the standard directories. **cc** passes the **-I** flag to the preprocessor.

7. To optimize the object code and produce an assembler listing:

```
cc -S -O pgm.c
```

This uses the optimizing compiler (**-O** is minus, capital oh), and produces an assembler listing in a file named `pgm.s` (**-S**).

## Files

<i>file.c</i>	C source file.
<i>file.o</i>	Object file.
<i>file.s</i>	Assembler file.
<i>a.out</i>	Linked output.
<i>/etc/cc.cfg</i>	<b>cc</b> configuration file.
<i>/tmp/ctm*</i>	Temporary.
<i>/lib/cpp</i>	C preprocessor.
<i>/lib/ccom0</i>	Compiler first pass.
<i>/lib/ccomq</i>	Intermediate code optimizer.
<i>/lib/ccom1</i>	Compiler second pass.
<i>/lib/cgen</i>	Compiler.
<i>/lib/copt</i>	optimizer.
<i>/bin/as</i>	Assembler.
<i>/bin/ld</i>	Linkage editor.
<i>/lib/crt0.o</i>	Runtime startoff.
<i>/lib/mcrt0.o</i>	Runtime startoff for profiling.
<i>/lib/libc.a</i>	Standard library.
<i>/lib/libfc.a</i>	Standard library for use with Floating-Point Accelerator.
<i>/lib/libm.a</i>	Standard math library.
<i>/lib/libfm.a</i>	Standard math library for use with Floating-Point Accelerator.
<i>/lib/librts.a</i>	Runtime services.
<i>/usr/include</i>	Standard directory for <b>#include</b> files.
<i>/usr/tmp/ctm*</i>	Temporary.

## Related Information

The following commands: “**as**” on page 64, “**ld**” on page 427, “**cpp**” on page 163, “**prof**” on page 571, and “**sdb**” on page 619.

The discussion of **cc** in *AIX Operating System Programming Tools and Interfaces*, in *C Language Guide and Reference* and in *Assembler Language Reference*.

The **monitor** subroutine and the **a.out** and **cc.cfg** files in *AIX Operating System Technical Reference*.



---

# cd

---

## Purpose

Changes the current directory.

## Syntax

```
cd directory
cd $HOME
```

OL805087

## Description

The **cd** command moves you from your present directory to another. You must have execute (search) permission in the specified *directory*.

If you do not specify a *directory*, **cd** moves you to your login directory (**\$HOME**). If the specified *directory* name is a full path name, it becomes the current directory. A full path name begins with a / (slash—root directory), with a . (dot—current directory), or with a .. (dot dot—parent directory). If the directory name is not a full path name, **cd** searches for it relative to one of the paths specified by the **\$CDPATH** shell variable. This variable has the same syntax as, and similar semantics to, the **\$PATH** shell variable. (See “Shell Variables and Command-Line Substitutions” on page 641 for a discussion of these variables.)

## Examples

1. To change to your home directory:  
cd
2. To change to an arbitrary directory:  
cd /usr/include

This changes the current directory to /usr/include. Now file path names that do not begin with / or ../ specify files located in /usr/include.

## cd

---

3. To go down one level of the directory tree:

```
cd sys
```

If the current directory is `/usr/include` and if it contains a subdirectory named `sys`, then `/usr/include/sys` becomes the current directory.

4. To go up one level of the directory tree:

```
cd ..
```

The special file name `..` (dot-dot) always refers to the directory immediately above the current directory.

## Related Information

The following commands: “**pwd**” on page 589 and “**sh**” on page 637.

The **chdir** system call in *AIX Operating System Technical Reference*.

---

# cdc

---

## Purpose

Changes the comments in a Source Code Control System (SCCS) delta.

## Syntax

```

cdc -rSID [-mmrlist] [-ycomment] file
cdc -rSID [-m] [-mmrlist] [-y] [-ycomment] -
  
```

OL805088

## Description

The **cdc** command changes the *Modification Requests* (MRs) and comments for the *SID* specified by the **-r** flag for each named *Source Code Control System* (SCCS) *file*. If you specify a directory name, **cdc** performs the requested actions on all SCCS files in that directory (that is, all files with names that have the *s.* prefix). If you specify a **-** (minus) in place of *file*, **cdc** reads standard input and interprets each line as the name of an SCCS file. For more information on SCCS comments and Modification Requests, see *AIX Operating System Programming Tools and Interfaces*.

You can change the comments and MRs for an SID only if you made the SID or you own the file and the directory. For more information on the permissions needed to change SCCS files, see “SCCS Files” on page 360.

## Flags

**-m**[*mrlist*] Supplies a list of MR numbers for **cdc** to add or delete in the SID specified by the **-r** flag. You can only use this flag if the *file* has the **v** header flag set (see Figure 1 on page 54). A null MR list has no effect.

In the *mrlist*, MRs are separated by blanks, tab characters, or both. To delete an MR, precede the MR number with an **!** (exclamation point). If the MR you want to delete is currently in the list of MRs, it is changed into a comment line. **cdc** places a list of all deleted MRs in the comment section of



the delta and precedes them with a comment line indicating that the following MRs were deleted.

If you do not specify the **-m** flag, and the **v** header flag is set, MRs are read from standard input. If standard input is a work station, **cdc** prompts you for the MRs. The first new-line character not preceded by a backslash ends the list on the command line. **cdc** continues to take input until it reads an end-of-file character (**Ctrl-D**) or a blank line. MRs are always read before comments (see the **-y** flag).

If the **v** flag has a value, **cdc** interprets the value as the name of a program which validates the MR numbers. If the MR number validation program returns a nonzero exit value, **cdc** stops and does not change the MRs.

- rSID** Specifies the SCCS identification number of the delta for which **cdc** will change the comments or MRs.
- y[comment]** Specifies text to replace any *comment* already existing for the delta specified by the **-r** flag. **cdc** keeps the existing comments and precedes them by a comment line stating that they were changed. A null *comment* has no effect.

If you do not specify **-y**, **cdc** reads comments from standard input until it reads an end-of-file character. If the standard input is a work station, **cdc** prompts for the comments and also allows a blank line to end input. If the last character of a line is a backslash (**\**), **cdc** ignores it and continues to read standard input.

**Note:** If **cdc** reads standard input for file names (that is, when you specify a file name of **-**), you must use the **-y** and **-m** flags.

## Related Information

The following commands: “**admin**” on page 51, “**delta**” on page 236, “**get**” on page 359, “**help**” on page 391, and “**prs**” on page 574.

The **sccsfile** file in *AIX Operating System Technical Reference*.

The discussion of SCCS in *AIX Operating System Programming Tools and Interfaces*.

---

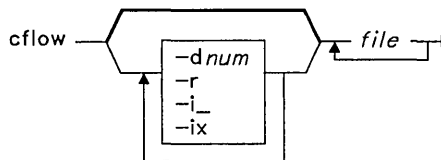
# cflow

---

## Purpose

Generates a C flow graph of external references.

## Syntax



OL805172

## Description

The **cflow** command analyzes C, **yacc**, **lex**, assembler, and object *files* and writes a chart of their external references to standard output.

It sends files with suffixes **.y**, **.l**, and **.c** to the **yacc**, **lex**, and **cpp** commands for the appropriate processing. This step is bypassed for **.i** files. It then runs the output of this processing through the first pass of **lint**. It assembles files which end in **.s**, extracting information from the symbol table (as it does with **.o** files). From this output, **cflow** produces a graph of external references, which it writes to standard output.

Each line of output begins with a line number followed by sufficient tabs to indicate the level of nesting. Then comes the name of the global, a colon, and its definition. This name is normally a function not defined as external and not beginning with an underline character; see the **-i\_** inclusion flag on p. 126. For information extracted from C source files, the definition consists of an abstract type declaration (for example, **char\***), the name of the source file, surrounded by angle brackets, and the line number on which the definition was found. Definitions extracted from object files contain the file name and location counter under which the symbol appeared. **cflow** deletes leading underline characters in C-style external names.

Once **cflow** displays the definition of a name, later references to it contain only the **cflow** line number where the definition may be found. For undefined references, **cflow** displays only **< >**.

If the nesting level becomes too deep to display in available space, pipe the output from **cflow** to the **pr** command, using the **-e** flag to compress the tab expansion to something less than every eight spaces.

## cflow

---

**Note:** Files produced by **lex** and **yacc** cause the reordering of line number declarations which can confuse **cflow**. To get proper results, feed **cflow** the **yacc** or **lex** input.

### Flags

In addition to the following, **cflow** recognizes the **-I**, **-D**, and **-U** flags of the **cpp** command.

- dnum** Sets to decimal integer *num* the depth at which the flow graph is cut off. By default this is a very large number. Do not set the cutoff depth to a nonpositive integer.
- ix** Includes external and static data symbols. The default includes only functions.
- i \_** Includes names that begin with an underline character. The default excludes these functions (and corresponding data if **-ix** is used).
- r** Produces an inverted listing which shows the callers of each function, sorted by called function.

### Related Information

The following commands: “**as**” on page 64, “**cc**” on page 112, “**lex**” on page 432, “**lint**” on page 446, “**nm**” on page 521, “**pr**” on page 561, and “**yacc**” on page 861.

The discussion of **cflow** in *AIX Operating System Programming Tools and Interfaces*.

---

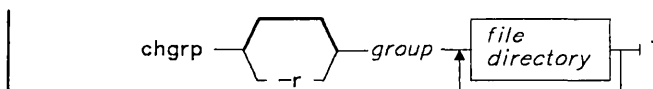
# chgrp

---

## Purpose

Changes the group ownership of a file or directory.

## Syntax



OL805090

## Description

The **chgrp** command changes the group associated with the specified *file* or *directory* to *groupname* or *groupID*. If you do not own the file, you must have superuser authority to change the group ID.

If the file or directory resides on a remote node, the translated group ID is used.

## Flag

**-r** Causes the untranslated group ID to be used.

## Examples

To change the group ownership of the file or directory named `proposals` to `staff`:

```
chgrp staff proposals
```

The group access permissions for `proposals` now apply to the `staff` group.

## Files

`/etc/group`

## **chgrp**

---

### **Related Information**

The following command: “**groups**” on page 385.

The **chown** and **chownx** system calls and the **group** file in *AIX Operating System Technical Reference*.

“Distributed Services **id** Translation” in *Managing the AIX Operating System*.



# chmod

---

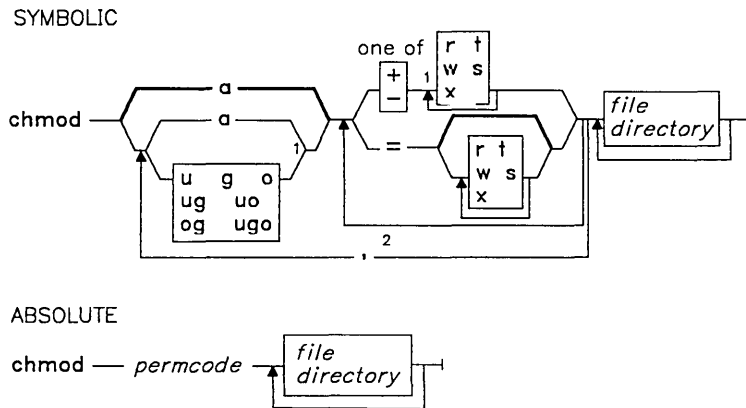
## chmod

---

### Purpose

Changes permission codes.

### Syntax



<sup>1</sup> Do not put a blank between these items.

<sup>2</sup> Do not put a blank on either side of the comma.

OL805091

### Description

The **chmod** command modifies the read, write, execute (*file*), or search (*directory*) permission codes of specified files or directories. You can use either symbolic or absolute mode to specify the desired permission settings.

You can change the permission code of a file or directory only if you own it or if you are operating with superuser authority.

---

## Symbolic Mode

When you use the symbolic mode to specify permission codes, the first set of flags selects the permission field, as follows:

<b>u</b>	User (owner)
<b>g</b>	Group
<b>o</b>	All others
<b>a</b>	User, group, and all others (same effect as <b>ugo</b> ). This is the default permission field.

The second set of flags selects whether permissions are to be taken away, added, or set exactly as specified:

<b>-</b>	Removes specified permissions
<b>+</b>	Adds specified permissions
<b>=</b>	Clears the selected permission field and sets it to the code specified. If you do not specify a permission code following <b>=</b> , <b>chmod</b> removes all permissions from the selected field.

The third set of flags of the **chmod** command selects the permissions as follows:

<b>r</b>	Read permission.
<b>w</b>	Write permission.
<b>x</b>	Execute permission for files; search permission for directories.
<b>s</b>	Set user-ID or set group-ID permission. This permission bit sets the effective user-ID or group-ID to that of the <i>file</i> whenever the <i>file</i> is run. Use this permission setting in combination with the <b>u</b> or <b>g</b> field to allow temporary or restricted access to files not normally accessible to other users. An <b>s</b> appears in the user or group execute position of a long listing (see “ <b>ls</b> ” on page 461 or “ <b>li</b> ” on page 437), to show that the file runs “set user-ID” or “set group-ID.”
<b>t</b>	The save text permission. Setting this permission bit causes the text segment of a program to remain in virtual memory after its first use. The system thus avoids having to transfer the program code of frequently-accessed programs into the paging area. A character special file with this bit set is a multiplexed file. You can specify this permission only with the <b>u</b> field. A <b>t</b> appears in the execute position of the “all others” field to indicate that the file has this bit (the <i>sticky</i> bit) set.

You can specify multiple symbolic modes, separated with commas. Do not separate items in this list with spaces. Operations are performed in the order they appear from left to right.



## Absolute Mode

The **chmod** command also permits you to use octal notation to set each bit in the permission code. **chmod** sets the permissions to the *permcode* you provide. This *permcode* is constructed by combining (the logical OR of) the following values:

<b>4000</b>	Sets user-ID on execution
<b>2000</b>	Sets group-ID on execution
<b>1000</b>	Retains memory image after execution (executable file)
<b>1000</b>	Indicates multiplexed character special file
<b>0400</b>	Permits read by owner
<b>0200</b>	Permits write by owner
<b>0100</b>	Permits execute or search by owner
<b>0040</b>	Permits read by group
<b>0020</b>	Permits write by group
<b>0010</b>	Permits execute or search by group
<b>0004</b>	Permits read by others
<b>0002</b>	Permits write by others
<b>0001</b>	Permits execute or search by others

All permission bits not explicitly specified are cleared.

## Examples

1. To add a type of permission to several files:

```
chmod g+w chap1 chap2
```

This adds write permission for group members to the files `chap1` and `chap2`.

2. To make several permission changes at once:

```
chmod go-w+x mydir
```

This denies group members and others the permission to create or delete files in `mydir` (`go-w`). It allows them to search `mydir` or use it in a path name (`go+x`). This is equivalent to the command sequence:

```
chmod g-w mydir
chmod o-w mydir
chmod g+x mydir
chmod o+x mydir
```

3. To permit only the owner to use a shell procedure as a command:

```
chmod u=rwx,go= cmd
```

This gives read, write, and execute permission to the user who owns the file (`u=rwx`). It also denies the group and others the permission to access `cmd` in any way (`go=`).

If you have permission to execute the shell command file `cmd`, then you can run it by entering:

```
cmd
```

This may not work in some cases, depending on the value of the shell variable `PATH`. See page 646 for more information about `PATH`.

4. To use “set-ID” modes:

```
chmod ug+s cmd
```

When `cmd` is executed, this causes the effective user and group IDs to be set to those that own the file `cmd`. Only the effective IDs associated with the subprocess that runs `cmd` are changed. The effective IDs of the shell session remain unchanged.

This feature allows you to permit restricted access to important files. Suppose that the file `cmd` has the set-user-ID mode enabled and is owned by a user called `dbms`. `dbms` is not actually a person, but might be associated with a database management system. The user `betty` does not have permission to access any of `dbms`’s data files. However, she does have permission to execute `cmd`. When she does so, her effective user ID is temporarily changed to `dbms`, so that the `cmd` program can access the data files owned by `dbms`.

This way `betty` can use `cmd` to access the data files, but she cannot accidentally damage them with the standard shell commands.

5. To use the absolute mode form of the `chmod` command:

```
chmod 644 text
```

This sets read and write permission for the owner, and it sets read-only mode for the group and others.

## Related Information

The following commands: “`ls`” on page 461, “`li`” on page 437, and “`umask`” on page 784.1.

**chown**

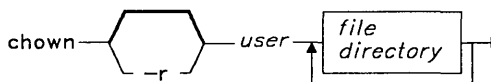
---

**chown**

---

**Purpose**

Changes the owner of files or directories.

**Syntax**

OL805095

**Description**

The **chown** command changes the owner of the specified *files* or *directories* to *username* or *userID*. The group associated with the file or directory is not affected.

**Note:** If you give ownership of a file or directory to another user, you cannot regain ownership unless you have superuser authority.

If the file or directory resides on a remote node, the translated user ID is used.

**Flag**

**-r** Causes the untranslated user ID to be used.

**Example**

```
chown jim program.c
```

The user access permissions for `program.c` now apply to `jim`. As the owner, `jim` can use **chmod** to permit or deny the other users access to `program.c`. See “**chmod**” on page 128 for details.

**Files**

`/etc/passwd`

## Related Information

The following command: “**passwd**” on page 546.

The **chown** and **chownx** system calls and the **passwd** file in *AIX Operating System Technical Reference*.

“Distributed Services **id** Translation” in *Managing the AIX Operating System*.

## **chown**

---

---

# chparm

---

## Purpose

Changes or examines system parameters.

## Syntax

```
chparm { nodename | nodename=newvalue } { /unix | kernel-image }
```

OL805093

## Description

The **chparm** command lets you change a system parameter or look at its current setting. Currently, only the **nodename** parameter may be examined or changed. The name assigned cannot be longer than eight characters. If you do not assign a *newvalue*, **chparm** writes the current value of **nodename** to standard output. The default *kernel-image* is **/unix**.

Changes do not affect the running system. You must restart the system for the change to become effective.

## Examples

1. To display the **nodename** of your system:

```
chparm nodename
```

This displays the **nodename** of **/unix**, which is a file containing the kernel of the AIX operating system. This file is loaded and run when you start up the computer.

2. To change the **nodename** of a system:

```
chparm nodename=COMP-CTR /unix.compctr
```

This changes the **nodename** of **/unix.compctr** to **COMP-CTR**. **/unix.compctr** is a file that contains an alternate version of the operating system kernel. The change does not affect the running system, even if you change the **/unix** kernel.

# chroot

---

## chroot

---

### Purpose

Changes the root directory of a command.

### Syntax

`chroot — directory — command —|`

OL805094

### Description

**Warning:** If special files in the new root have different major and minor device numbers than they have in the real root, it is possible to overwrite the file system.

The **chroot** command can be used only by a user operating with superuser authority (see “su” on page 724). If you have superuser authority, the **chroot** command changes the root directory to the specified *directory* when executing *command*. The first / (slash) in any path name changes to *directory* for the specified *command* and any of its children.

Notice that:

```
chroot directory command > file
```

creates the *file*. relative to the original root, not the new one.

The *directory* path name is always relative to the current root. Even if a **chroot** is in effect, *directory* is relative to the current root of the running process.

Several programs may not operate properly after **chroot** has been run. For example, the command `ls -l` will fail to give user and group names if the current root location makes `/etc/passwd` beyond reach. In addition, utilities that depend on description files produced by the **ctab** command (see page 204) may fail altogether if these files are also not in the new root file system. It is your responsibility to ensure that all vital data files are present in the new root file system and that the path names accessing such files are changed as necessary.

## Examples

1. To run a subshell with another file system as the root:

```
chroot /diskette0 /bin/sh
```

This makes the directory name `/` refer to `/diskette0` for the duration of the command `/bin/sh`. It also makes the original root file system inaccessible. The file system on `/diskette0` must contain the standard directories of a root file system. In particular, the shell will look for commands in `/bin` and `/usr/bin` on the `/diskette0` file system.

Running the command `/bin/sh` creates a subshell, which runs as a separate process from your original shell. Press END OF FILE (**Ctrl-D**) to end the subshell and go back to where you were in the original shell. This restores the environment of the original shell, including the meanings of the current directory (`.`) and the root directory (`/`).

2. To run a command in another root file system and save the output:

```
chroot /diskette0 /bin/cc -E /u/bob/prog.c >prep.out
```

This runs the `/bin/cc` command with `/` referring to `/diskette0`. It saves the output in the file `prep.out`, which is in the original root file system.

This runs the C language preprocessor (`/bin/cc -E`) on the file `/diskette0/u/bob/prog.c`, reading `#include` files from `/diskette0/usr/include`, and putting the preprocessed text in `prep.out` on the primary root file system.

## Related Information

The following commands: “**cc**” on page 112, “**cpp**” on page 163, and “**sh**” on page 637.

The `chdir` and `chroot` system calls in *AIX Operating System Technical Reference*.



# clri

---

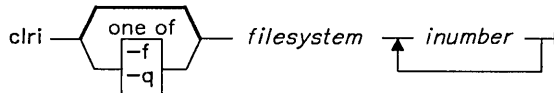
## clri

---

### Purpose

Clears the specified i-node.

### Syntax



OL805097

### Description

**Warning:** Use this command only in emergencies and with extreme care.

The **clri** command is used to clear i-node entries for files that do not appear in a directory. In general, you do not need to use this program because **fsck** can deal with most file system inconsistencies.

Always run **fsck** on a file system after you have used **clri** on it, because it may create dangling directory references or missing blocks. These can be fixed if they are attended to promptly. Do not run the system when the file system has dangling directory references or a bad free list.

The **clri** command zeroes over the flags word of the i-node, thus freeing it for reallocation. The *inumber* parameter specifies the i-node and *filesystem* specifies the file system it is on. *inumber* should be a decimal number, while *filesystem* can be either the name of the device on which the file system resides or the name by which it is normally mounted.

If you use **clri** to remove an i-node that does appear in a directory, you should track down and remove all of these entries. Otherwise, when the i-node is reallocated to some new file, the old entry will still point to that file. At that point removing the old entry destroys the new file and the new entry again points to an unallocated i-node.

By default, the **clri** command displays some information about the file and asks for confirmation before it destroys the file. If you enter a *y* or *yes*, the file is destroyed.

Since **clri** only zeroes the flags word of the i-node, if you destroy the wrong file, you can recover the file by using the **fsdb** command to restore the flags word.

**Note:** If the file is open, **clri** is likely to be ineffective. For this reason, you should run **clri** only on an unmounted file system.

## Flags

- f Destroys the file without confirmation, but writes a description of the file.
- q Destroys the file without confirmation or writing a description of the file.

## Example

To clear i-nodes 170 and 368 of the file system `/diskette0` and then clean up the file system:

```
clri /diskette0 170 368
fsck /diskette0
```

## Related Information

The following commands: “**fsck**, **dfsc**” on page 333 and “**fsdb**” on page 338.

The **fs** file in *AIX Operating System Technical Reference*.

# cmp

---

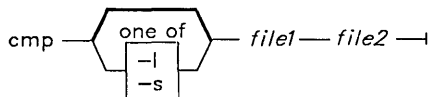
## cmp

---

### Purpose

Compares two files.

### Syntax



OL805157

### Description

The **cmp** command compares *file1* and *file2* and writes the results to standard output. If you specify a - (minus) for *file1*, **cmp** reads standard input. Under default conditions, **cmp** displays nothing if the files are the same. If they differ, **cmp** displays the byte and line number at which the first difference occurs. If one file is an initial subsequence of the other (that is, if **cmp** reads an end-of-file character in one file before finding any differences), **cmp** notes this. Normally, you use **cmp** to compare non-text files and the **diff** command to compare text files.

### Flags

- l Displays, for each difference, the byte number in decimal and the differing bytes in octal.
- s Returns only an exit value. (0 indicates identical files; 1 indicates different files; 2 indicates inaccessible file or a missing argument)

### Examples

1. To determine whether two files are identical:

```
cmp prog.o.bak prog.o
```

This compares *prog.o.bak* and *prog.o*. If the files are identical, then a message is not displayed. If the files differ, then the location of the first difference is displayed.

For instance:

```
prog.o.bak prog.o differ: char 5, line 1
```

If the message `cmp: EOF on prog.o.bak` is displayed, then the first part of `prog.o` is identical to `prog.o.bak`, but there is additional data in `prog.o`.

2. To display each pair of bytes that differ:

```
cmp -l prog.o.bak prog.o
```

This compares the files, and then displays the byte number (in decimal) and the differing bytes (in octal) for each difference. For example, if the fifth byte is octal 101 in `prog.o.bak` and 141 in `prog.o`, then `cmp` displays:

```
5 101 141
```

3. To compare two files without writing any messages:

```
cmp -s prog.c.bak prog.c
```

This gives an exit value of 0 if the files are identical, 1 if different, or 2 if an error occurs. This form of the command is normally used in shell procedures. For example:

```
if cmp -s prog.c.bak prog.c
then
    echo No change
fi
```

This partial shell procedure displays `No change` if the two files are identical. See page 653 for details about the `if` command.

## Related Information

The following commands: “**comm**” on page 144, “**diff**” on page 246, and “**sh**” on page 637.

# col

---

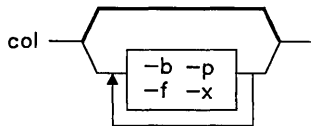
## col

---

### Purpose

Processes text having reverse linefeeds and forward/reverse half-linefeeds for output to standard output.

### Syntax



OL805173

### Description

The **col** command reads from standard input and writes to standard output. It performs the line overlays implied by reverse linefeeds (ASCII ESC-7), and by forward and reverse half-linefeeds (ASCII ESC-9 and ASCII ESC-8). **col** is particularly useful for filtering multi-column output made by the **nroff .rt** command and output from the **tbl** command. The input format accepted by **col** matches the output format produced by **nroff -T37** or by **nroff -Tlp**. Use **-T37** and the **col -f** flag if the output is being sent to a device that can interpret half-line motions; use **-Tlp** otherwise.

The **col** command assumes that the ASCII control characters SO (\017) and SI (\016) begin and end text in an alternate character set. **col** remembers the character set each input character belongs to, and on output generates SI and SO characters as appropriate to ensure that each character is printed in the correct character set.

On input, **col** accepts only the control characters for space, backspace, tab, return, the new-line character, SI, SO, VT, and ESC-7, 8, or 9. VT (\013) is an alternate form of full reverse linefeed included for compatibility with some earlier programs of this type. **col** ignores all other non-printing characters.

**Note:** The **col** command cannot back up more than 128 lines.

It allows at most 800 characters, including backspaces, on a line.

It ignores local vertical motions that would result in backing up over the first line. As a result, the first line must not contain any superscripts.

---

## Flags

- b Assumes that the output device in use is not capable of backspacing. In this case, if two or more characters are to appear in the same position, only the last one read appears in the output.
- f Suppresses the default treatment of half-line motions in the input. Normally, **col** does not emit half-line motions on output, although it does accept them in its input. With this flag, output may contain forward half-linefeeds (ESC-9) but not reverse linefeeds (ESC-7 or ESC-8).
- p Displays unknown escape sequences as characters, subject to overprinting from reverse line motions. Normally, **col** ignores them. You should be fully aware of the textual position of escape sequences before you use this flag.
- x Suppresses changing the white space to tabs. Without this flag, **col** converts white space to tabs wherever doing so might shorten printing time.

## Related Information

The following commands: “**nroff**” on page 525 and “**tbl**” on page 739.

The discussion of **col** in *Text Formatting Guide*.

# comb

---

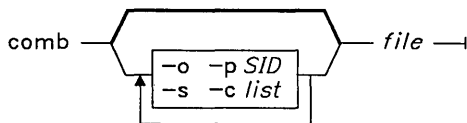
## comb

---

### Purpose

Combines SCCS deltas.

### Syntax



OL805098

### Description

The **comb** command writes to standard output a shell procedure that can combine the specified deltas (*SIDs*) or all deltas into one delta. You may reduce the size of your SCCS file by running the resulting procedure on the file. You can see how much the file will be reduced by running **comb** with the **-s** flag. If you specify a directory in place of *file*, **comb** performs the requested actions on all SCCS files (that is, those with file names with the **s.** prefix). If you specify a **-** (minus) in place of *file*, **comb** reads standard input and interprets each line as the name of an SCCS file. **comb** continues to take input until it reads END OF FILE (**Ctrl-D**).

If you do not specify any flags, **comb** preserves only leaf deltas and the minimal number of ancestors needed to preserve the tree (see “**delta**” on page 236).

**Note:** The **comb** command may rearrange the shape of the tree deltas. It may not save any space; in fact, it is possible for the reconstructed file to actually be larger than the original.

### Flags

Each flag or group of flags applies independently to each named file.

**-clist** Specifies a list of deltas (*SIDs*) that the shell procedure will preserve (see **get -i list** for the *SID* list format on page 364). The procedure will combine all other deltas.

- o**      Accesses the reconstructed file at the release of the delta to be created for each **get -e** generated; otherwise accesses the reconstructed file at the most recent ancestor. Using the **-o** flag may decrease the size of the reconstructed SCCS file. It may also alter the shape of the delta tree of the original file.
- pSID**    Specifies the *SID* of the oldest delta for the resulting procedure to preserve. All older deltas are combined in the reconstructed file.
- s**      Causes **comb** to generate a shell procedure that produces a report for each file giving: the file name, size (in blocks) after combining, original size (also in blocks), and percentage change computed by the formula:  
$$100 * (\text{original} - \text{combined}) / \text{original}$$

You should run **comb** using this flag and run its procedure before combining SCCS files in order to judge how much space will actually be saved by the combining process.

## Files

- s.COMB      The name of the reconstructed SCCS file.
- comb\*        Temporary files.

## Related Information

The following commands: “**admin**” on page 51, “**delta**” on page 236, “**get**” on page 359, “**help**” on page 391, and “**prs**” on page 574.

The **sccsfile** file in *AIX Operating System Technical Reference*.

The discussion of SCCS in *AIX Operating System Programming Tools and Interfaces*.



## comm

---

## comm

---

### Purpose

Selects or rejects lines common to two sorted files.

### Syntax

comm 

one of		
-1	-2	-3
-12	-13	-23
-123		

*file1* — *file2* —

OL805099

### Description

The **comm** command reads *file1* and *file2* and writes, by default, a three-column output to standard output. The columns consist of:

1. Lines that are only in *file1*
2. Lines that are only in *file2*
3. Lines that are in both *file1* and *file2*.

If you specify - (minus) for one of the file names, **comm** reads standard input. Both *file1* and *file2* should be sorted according to the collating sequence specified by the environment variable **NLCTAB** (see “**ctab**” on page 204 and “**sort**” on page 672),

### Flags

- 1 Suppresses the display of the first column (lines in *file1*).
- 2 Suppresses the display of the second column (lines in *file2*).
- 3 Suppresses the display of the third column (lines common to *file1* and *file2*).

**Note:** Specifying **-123** does nothing (a noop).

### Examples

1. To display the lines unique to each file and common to both:  
comm things.to.do things.done

If the files `things.to.do` and `things.done` contain:

<code>things.to.do</code>	<code>things.done</code>
buy soap	2nd revision
groceries	interview
luncheon	luncheon
meeting at 3	system update
system update	tech. review
tech. review	weekly report

then **comm** displays:

```

    2nd revision
buy soap
groceries
    interview
        luncheon
meeting at 3
        system update
        tech. review
        weekly report
```

The first column contains the lines found only in `things.to.do`. The second column, indented with a tab character, lists the lines found only in `things.done`. The third column, indented with two tabs, lists the lines common to both.

- To display the lines that appear in only one file:

```
comm -23 things.to.do things.done
```

This suppresses the second and third columns of the **comm** listing. If the files are the same as in Example 1, then the following is displayed:

```
buy soap
groceries
meeting at 3
```

## Related Information

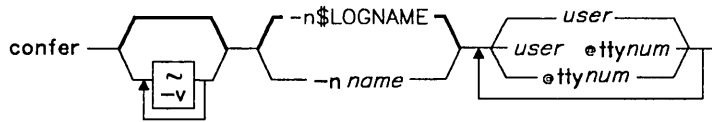
The following commands: “**cmp**” on page 138, “**ctab**” on page 204, “**diff**” on page 246, “**sdiff**” on page 627, “**sort**” on page 672, and “**uniq**” on page 792.

The **environment** miscellaneous facility in *AIX Operating System Technical Reference*.

“Overview of International Character Support” in *Managing the AIX Operating System*.

**confer****confer****Purpose**

Provides an on-line conferencing system.

**Syntax**

joinconf — *name* —

OL805174

**Description**

The **confer** command sets up an on-line, written conference among logged-in users on your local node. You start a conference by running the **confer** command, specifying the *users* and/or work stations (**@ttynum**) that are part of the conference. If the users you specify are logged in and their work stations are writable, they are requested to join the conference by using the **joinconf** command. The other conferees are informed as each user joins the conference.

Once you join a conference, everything you enter at your work station displays at all other work stations that are part of the conference. This display continues until you press **Ctrl-D** to end your own active participation or until you **excuse** a conference participant, thus stopping the display of your contributions at his work station. (See page 147.)

To prevent the confusion that can be caused by several conferees typing at the same time, users should follow some agreed on protocol. The following is one recommended protocol:

- In order to take the floor, a user presses the **Enter** key before entering his contribution. This notifies other participants that he has the floor because his name displays in brackets at their respective work stations.
- A user is presumed to have the floor until he relinquishes it by entering a blank line.
- If two or more users try to claim the floor at the same time, the last person to do so (the one whose name appears last), is assumed to have the floor. The others should immediately relinquish the floor by typing single blank lines.

The **confer** command gives each conference a unique name, normally the name of the conference leader, with additional letters added to it, if necessary. The conference leader can override this default by specifying the **-n** flag.

A user who is logged in to more than one work station is normally written to on all of them, unless the conference leader specifies one of the work stations with the **@ttynum** flag when he invokes **confer**.

A conferee ends his active participation by pressing **Ctrl-D**. This action causes his name and the word **BYE** to display at the work stations of the other conference participants. However, the contributions of the other participants will continue to display at his work station until the other participants each **excuse** him.

You can run shell commands from within a conference by simply prefixing them with a | (vertical bar) or an ! (exclamation point). Using the exclamation point causes the command to run in the normal fashion; the output displays only at the work station that runs it. Using the vertical bar, however, causes the command and all of its standard output and standard error output to become part of the conference, visible to all conferees.

Three subcommands are run directly by **confer** and **joinconf**. These are:

- !excuse name . . .** Excuses the specified conferees from the conference. No further conference material displays at these work stations.
- !~** Makes all contributions from the user who issues it off the record until he issues the **!~~** subcommand.
- !~~** Cancels a preceding **!~**, placing the user's remarks back on the record.

Unless the conference leader makes a conference off the record by specifying the **~** flag, **confer** makes a transcript of all conference proceedings. When a participant leaves the conference, he is asked whether he wants a transcript. If he does, he is mailed a copy when the conference concludes. Any participant can make a comment off the record in a conference that is otherwise **on the record** by beginning the line with a **~** (tilde).

Conference contributions are normally transmitted one line at a time. If the conference leader specifies the **-v** flag, transmission occurs one character at a time. As this mode of transmission sends all user typing errors and hesitations and imposes a considerably larger load on the system, its use is strongly discouraged.

## Flags

- nname** Assigns *name* to the conference transcript. The conference name is used by those joining the conference so that they get into the right one. The name of the user who starts the conference is the default conference name.
- v** Transmits conference messages one character at a time.
- ~** Sets up the conference **off the record**, that is, no transcript of the proceedings is recorded.

## confer

---

**@ttynum** Specifies a particular work station for a conferee, if a *user* is also specified (for example, `tty1`). This is useful if a conferee is logged in to more than one work station. If no *user* is specified, this flag invites any user logged in to the specified work station to participate.

### Examples

1. To start a conference with `steve` and `rachel`:

```
confer steve rachel
```

Running the **confer** command makes you the conference leader, so your login name is also the name of the conference. **confer** sends `steve` and `rachel` a message inviting them to join your conference and giving them the conference name.

2. To specify work stations that may join the conference:

```
confer steve@tty5 rachel @tty10
```

Suppose that `steve` is logged in at the work stations `tty3`, `tty4`, and `tty5`, and that `rachel` is logged in at `tty7` and `tty8`. This command invites `steve` to join the conference at work station `tty5` only, invites `rachel` to join at either work station she is using or at both, and invites whoever is logged in at `tty10` to join.

3. To join a conference named `paula`:

```
joinconf paula
```

Now the text you type becomes part of the dialog: prefixed with your name, displayed at each participant's work station, and recorded in the transcript of the conference.

4. Suppose that you start a conference by entering the command given in Example 2, and the person using `tty10` decides not to join the conference. If you do nothing, this person also sees the dialog, even though not participating in it. To prevent this from happening, each person that has joined the conference must enter:

```
!excuse @tty10
```

Similarly, if `rachel` decides to join the conference from `tty7`, the discussion is also displayed at her other work station, `tty8`, unless everyone enters:

```
!excuse rachel@tty8
```

**rachel** should enter this, too, but only at `tty7`, the work station she is using for the conference.

5. To make a single-line statement off the record:

```
~Coffee and donuts at my place.
```

**confer** displays lines beginning with `~` (tilde) at participants' work stations, but does not include them in the record of the conference.

To make a multiple-line statement:

```
!~
Everyone is invited
to my place after the conference
for coffee and donuts.
!~~
```

6. To run a shell command privately, without leaving the conference:

```
!!i
```

This lists the current directory without including the `li` command or its output in the conference.

7. To include the output of a shell command in the discussion:

```
|cat notes.conf
```

This lists the contents of the file `notes.conf` at each participant's work station, and includes it in the conference record.

8. To send command output to others, off the record:

```
!~
|cat notes.conf
!~~
```

9. To leave the conference, press **Ctrl-D**. If your user name is `paula`, then after you press **Ctrl-D**, the message: `[paula] BYE` is sent to the other participants. The rest of the discussion continues to appear at your work station until each of the other participants enters:

```
!excuse paula
```

## Files

<code>/etc/utmp</code>	List of logged-in users.
<code>/dev/tty??</code>	Work station names.
<code>/tmp/*.cnf</code>	User transcript files.
<code>/tmp/*.ln?</code>	Links to main conference file.
<code>/tmp/*.mls</code>	Transcript mailing list.

## Related Information

The following command: “**write**” on page 853.

# config

---

## config

---

### Purpose

Extracts configuration information from configuration files.

### Syntax

```
config -m /etc/master -c conf.c -l specials systemfile  
      -m mfile -c cfile -l spfile
```

OL805416

### Description

The **config** program reads the AIX master and system configuration files (by default **/etc/master** and the specified *systemfile*). It writes a C Language configuration file and a special file list (by default **conf.c** and **specials**). The special file list is a list of the **mknod**, **chown**, and **chmod** commands that the shell runs to define the necessary special files. The return code is the number of errors encountered.

The C Language configuration file can then be compiled and linked with other kernel object files to produce a new kernel. Normally, when you want to reconfigure the kernel, you should run the **make** command with the **Makefile** supplied in the **/usr/sys** directory. This runs **config** and then builds a new kernel. For a discussion of reconfiguring the kernel, see *Managing the AIX Operating System*.

### Flags

- c cfile**     Writes the C configuration file to *cfile* instead of to **conf.c**.
- l spfile**    Writes the special file list commands to *spfile* instead of to **specials**.
- m mfile**     Reads *mfile* instead of **/etc/master**.

### Files

<b>/etc/master</b>	Default master configuration file.
<b>/etc/system</b>	A system configuration file.
<b>conf.c</b>	Default C configuration file.
<b>specials</b>	Default special file list.

## **Related Information**

The following commands: “**make**” on page 474 and “**vrmsconfig**” on page 842.

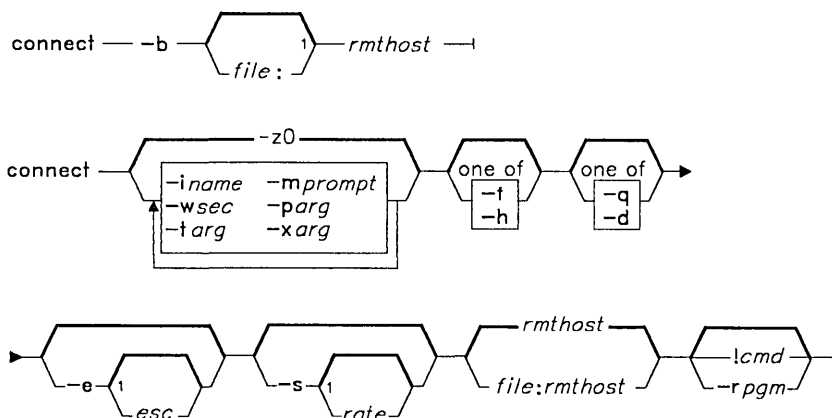
The **master** and **system** files in *AIX Operating System Technical Reference*.

The discussion of **config** in *Managing the AIX Operating System*.



**connect****connect****Purpose**

Establishes a connection to a remote system.

**Syntax**

OL805388

**Description**

The **connect** command lets you establish a connection to a remote host. **connect** runs in two parts. The first part makes the connection with the remote system specified by *rmthost*. The second part is a program called the **talker**. It runs automatically and exchanges data with the *rmthost*. For information about the **talker** program, see **connect** in *AIX Operating System Technical Reference*. Any flags that you specify are passed directly to the **talker** without interpretation. The default **talker** for asynchronous links is **atalk**.

The **connect** command uses a system-wide control file, **connect.con**, located in `/usr/lib/INnet`. You can specify an additional control file, *file:rmthost*. If you do not specify an additional file, **connect** searches `$HOME/bin` for a **connect.con** file. Information needed to complete the connection is found in one of these files.

Attributes needed to complete the connection are taken from the control file or from the command line assignment *var=val*. For a description of the parameters, see **connect** in *AIX Operating System Technical Reference*.

When **atalk** detects an escape sequence in the input, it places the work station in its former mode of operation and prompts you with the local prompt. You can then use the flags that follow. Once the flag has run, **atalk** returns to its former mode.

The **connect** command does not limit access to the phone system to control dialing based on the number to be called.

**Warning:** The **connect** command lets you set up and maintain connections through a wide variety of communications devices. It interacts with you through the file **connect.con** which is free-format. Problems with the format of this file may cause unpredictable results.

## Flags

**Note:** There are no spaces between the flags and the associated parameters.

**-b** Sends a break to the port. This is done by lowering the transmission speed to 75 bps and transmitting an ASCII NULL on the port. If the speed is too low, less than 100 bps, this may not work.

**-d**

**-q** Closes, quits (**q**) or disconnects (**d**) the port. Note that this does *not* end your job or session at the remote site. After closing the port, **connect** exits.

**-e[esc]**

Sets the escape sequence to the character string *esc*. If you do not specify *esc*, **connect** displays escape sequence. It takes the default escape sequence from the environment variable **CONESC**, if defined, or else sets it to:

**Ctrl-VuCtrl-M**

**-f**

**-h** Enables (**-h**) or disables (**-f**) local echoing.

**-iname**

Writes file *name* to the port.

**Warning:** If you are connected to the remote host by RS-232 lines, data from the file may be lost if the remote host cannot keep up with the input.

Normally, this flag is used to transfer a small file from the local site to the remote site. File transmission must be ended manually by pressing **Ctrl-D**.

## connect

---

For example:

```
cat > newfile  
[escape sequence]  
LOCAL: ifred
```

```
.  
. .  
.
```

Ctrl-D

- mprompt** Set the local prompt to the *prompt* character string. **connect** displays this prompt when it recognizes the escape sequence. By default, it sets the prompt to the value of the environment variable **CONPMT**. If this variable is not set, it uses the the string **LOCAL:**.
- parg** Sets parity as specified by *arg*, where *arg* is one of the following characters: **o** (odd), **e** (even), **7** (both even and odd), or **8** (eight data bits).
- rpgm** Runs the network program *pgm*. Anything following *pgm* on the command line is passed to *pgm* as an argument, along with the additional arguments **-i3 -o3**. The port set up as file descriptor 3. The program is run as a child process.
- srate** Sets the transmission speed to *rate*, which is one of the following: 0, 50, 75, 110, 134, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, exta, extb (0 effectively turns off the port). If you do not specify *rate*, current transmission speed displays.
- targ** Enables or disables transcripts. If *arg* is any character string other than a minus or plus sign, the transcript function is enabled with the specified file *arg* as transcript. When you use an existing file as a transcript file, new data is added to its end. Use **t-** to disable the transcript function, and **t+** to enable the transcript to the previous transcript file (no default).
- wsec** Sets the inter-line delay of the include function to cause a delay interval of the specified seconds between each line written to the port. The default value is 0.
- xarg** Enables or disables input or output flow control. If the input flow control is enabled, **CTRL-S** and **CTRL-Q** are automatically sent to the remote host to control the rate at which it transmits data. If the output flow control is enabled, **CTRL-S** and **CTRL-Q** are automatically honored if received from the host. This is useful when using the **include** command. **xi+** enables input flow control. **xi-** disables input flow control. **xi** displays the current state. For control of output flow control, replace **xi** with **xo**. See the discussion of IXON and INOFF in the **termio** file in *AIX Operating System Technical Reference*.
- !cmd** Runs the AIX command *cmd*. Anything that follows **!**, including arguments to *cmd*, is passed to the local shell to be run by the **system** system call. In particular, all I/O redirection and piping works.

## Files

/usr/lib/INnet/connect.con	System-wide connection control file.
\$HOME/bin/connect.con	Private connection control file.
/usr/lib/INnet/dialers/*	System-wide dialer programs.
\$HOME/bin/*	Private dialer programs.
/usr/lib/INnet/atalc	Default talker program, asynchronous lines.
/etc/sites	Network sites file.
/etc/locks	Directory for locks on ports (devices) used for logins and out-going connections.

## Related Information

The **system** and **exec** system calls, the **connect** subroutine, and the **termio** special facility in *AIX Operating System Technical Reference*.

## cp

---

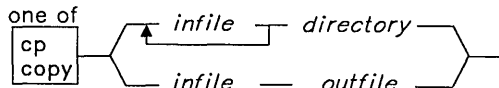
## cp

---

### Purpose

Copies files.

### Syntax



OL805100

### Description

The **cp** (**copy**) command copies *infile* to a *directory* or another file, *outfile*. Do not name *outfile* as one of the input files. If your output is to a *directory*, then the files are copied to that directory with the same base file name.

You can also copy special device files. If the file is a named pipe, the data in the pipe is copied into a regular file. If the file is a device, the file is read until the end of file and that data is copied into a regular file.

### Examples

1. To make another copy of a file in the current directory:

```
cp prog.c prog.bak
```

This copies `prog.c` to `prog.bak`. If the file `prog.bak` does not already exist, then **cp** creates it. If it does exist, then **cp** replaces it with a copy of `prog.c`.

2. To copy a file to the same name in another directory:

```
cp jones clients
```

This copies `jones` to `clients/jones`.

**Note the difference:** `prog.bak` in Example 1 is the name of a file; `clients` in Example 2 is a directory that already exists.

3. To copy several files into another directory:

```
cp listing clients/smith /u/tom
```

This copies `listing` to `/u/tom/listing` and `clients/smith` to `/u/tom/smith`.

4. To use **cp** with pattern-matching characters:

```
cp programs/*.c .
```

This copies all of the files in directory `programs` that end with `.c` into the current directory (`.`), giving them the same names they have in `programs`. Note that you must type a space between the `c` and the final period.

## Related Information

The following commands: “**cpio**” on page 158, “**link, unlink**” on page 444, “**ln**” on page 450, and “**mv**” on page 502.

# cpio

---

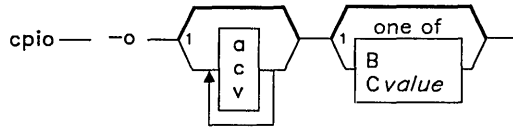
# cpio

---

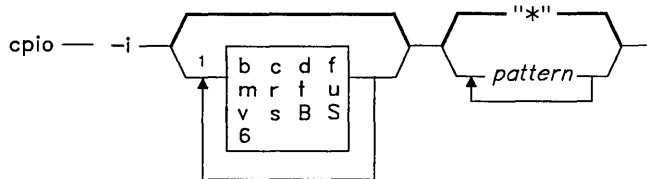
## Purpose

Copies files into and out of archive storage and directories.

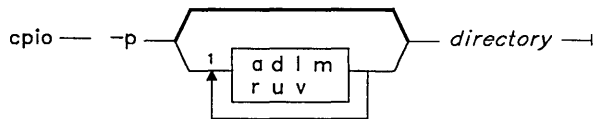
## Syntax



OL805175



OL805350



<sup>1</sup>Do not put a blank between these items.

OL805351

## Description

**Warning:** If you redirect the output from `cpio` to a special file (device), you should redirect it to the raw device and not the block device. Because writing to a block device is done asynchronously, there is no way to know if the end of the device has been reached.

## **cpio -o**

This command reads file path names from standard input and copies these files to standard output along with path names and status information. Path names cannot exceed 128 characters. Avoid giving **cpio** path names made up of many unique linked files as it may not have enough memory to keep track of them and so would lose linking information.

## **cpio -i**

This command reads from standard input the name of an archive file created by the **cpio -o** command and copies from it the files with names that match *pattern*. These files are copied into the current directory tree. You may list more than one *pattern*, using the file name notation described under “**sh**” on page 637. Note, however, that in this application the special characters \*, ?, and [ . . . ] match the / (slash) in path names, in addition to their use as described under “**sh**” on page 637. The default *pattern* is \* (select all files in the current directory).

In an expression such as [a-z], the minus means “through” according to the current collating sequence. A collating sequence may define *equivalence classes* for use in character ranges. See the “Overview of International Character Support” in *Managing the AIX Operating System* for more information on collating sequences and equivalence classes.

## **cpio -p**

This command reads file path names from standard input and copies these files into the named *directory*. The specified *directory* must already exist. If these path names include directory names and if these directories do not already exist, you must use the **d** flag to cause the *directory* to be created.

**Note:** You can copy special files only if you have superuser authority.

## **Flags**

All flags must be listed together, without any blanks between them. Not all of the following flags can be used with each of the **-o**, **-i**, and **-p** flags.

**a** Resets the access times of copied files to the current time.

**b** Swaps both bytes and halfwords.

**Note:** If there are an odd number of bytes or halfwords in the file being processed, data can be lost.

**B** Performs block input/output, 5120 bytes to a record.

**c** Writes header information in ASCII character form.



## cpio

---

- Cvalue** Performs block input/output, *value* \* 512 bytes to a record.
- Note:** The **C** flag and the **B** flag are mutually exclusive. If you list both, **cpio** uses the last one it encounters in the flag list.
- d** Creates directories as needed.
- f** Copies all files except those matching *pattern*.
- l** Links files rather than copies them, whenever possible. This flag is usable only with **cpio -p**.
- m** Retains previous file modification time. This flag does not work when copying directories.
- r** Renames files interactively. If you do not want to change the file name, enter the current file name or press the **Enter** key only. In this last case, **cpio** does not copy the file.
- s** Swaps bytes. This flag is usable only with **cpio -i**.
- Note:** If there are an odd number of bytes in the file being processed, data can be lost.
- S** Swaps halfwords. This flag is usable only with **cpio -i**.
- Note:** If there are an odd number of halfwords in the file being processed, data can be lost.
- t** Creates a table of contents. This does not copy any files.
- u** Copies unconditionally. An older file now replaces a newer file with the same name.
- v** Lists file names. If you use this with the **t** flag, the output looks similar to that of the **ls -l** command.
- 6** Processes an old file (one written in UNIX Sixth Edition format). This flag is usable only with **cpio -i**.

## Examples

1. To copy files onto diskette:

```
cpio -ov <filenames >/dev/rfd0
```

This copies the files with path names that are listed in the file *filenames* in a compact form onto the diskette (*>/dev/rfd0*). The **-v** flag causes **cpio** to display the name of each file as it is copied. This command is useful for making backup copies of files. The diskette must already be formatted, but it must not contain a file system or be mounted.

2. To copy files in the current directory onto diskette:

```
ls *.c | cpio -ov >/dev/rfd0
```

This copies all the files in the current directory whose names end with `.c`.

3. To copy the current directory and all subdirectories onto diskette:

```
find . -print | cpio -ov >/dev/rfd0
```

This saves the directory tree that starts with the current directory (`.`) and includes all of its subdirectories and files. A faster way to do this is:

```
find . -cpio /dev/rfd0 -print
```

The `-print` displays the name of each file as it is copied.

4. To list the files that have been saved onto a diskette with **cpio**:

```
cpio -itv </dev/rfd0
```

This displays the table of contents of the data previously saved onto `/dev/rfd0` in **cpio** format. The listing is similar to the long directory listing produced by `li -l`. To list only the file path names, use only the `-it` flags.

5. To copy the files previously saved with **cpio** from a diskette:

```
cpio -idmv </dev/rfd0
```

This copies the files previously saved onto `/dev/rfd0` by **cpio** back into (`-i`) the file system. The `-d` flag allows **cpio** to create the appropriate directories if a directory tree was saved. The `-m` flag maintains the last modification time that was in effect when the files were saved. The `-v` causes **cpio** to display the name of each file as it is copied.

6. To copy selected files from diskette:

```
cpio -i "*.c" "*.o" </dev/rfd0
```

This copies the files that end with `.c` or `.o` from diskette. Note that the patterns `"*.c"` and `"*.o"` must be enclosed in quotation marks to prevent the shell from treating the `*` as a pattern-matching character. This is a special case in which **cpio** itself decodes the pattern-matching characters.

7. To rename files as they are copied from diskette:

```
cpio -ir </dev/rfd0
```

The `-r` flag causes **cpio** to ask you whether or not to rename each file before copying it from diskette. For example, the message:

```
Rename <prog.c>
```

asks whether to give the file saved as `prog.c` a new name as it is copied in. To rename the file, type the new name and press **Enter**. To keep the same name, you must enter the name again. To avoid copying the file at all, press the **Enter** key alone.

## **cpio**

---

8. To copy a directory and all of its subdirectories:

```
mkdir /u/jim/newdir  
find . -print | cpio -pdl /u/jim/newdir
```

This duplicates the current directory tree, including the current directory and all of its subdirectories and files. The duplicate is placed in the new directory `/u/jim/newdir`. The `-l` flag causes **cpio** to link files instead of copying them, when possible.

### **Related Information**

The following commands: “**ar**” on page 58, “**find**” on page 326, and “**ln**” on page 450.

The **cpio** system call in *AIX Operating System Technical Reference*.

“Overview of International Character Support” in *Managing the AIX Operating System*.

---

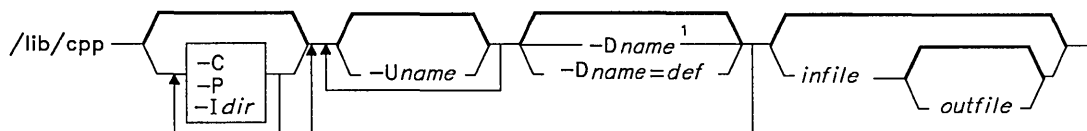
# cpp

---

## Purpose

Performs file inclusion and macro substitution on C Language source files.

## Syntax



<sup>1</sup>The default *def* is 1.

OL805378

## Description

The `cpp` program is the C Language preprocessor. It reads *infile* and writes to *outfile* (standard input and standard output by default). Although you can use this preprocessor by itself, it is best to use it through the `cc` command, which by default sends a C Language source file to `cpp` as the first pass in compilation.

The `cpp` program recognizes two special names, `__LINE__` (the current line number) and `__FILE__` (current file name). These names can be used anywhere just as any other defined name.

All `cpp` directive lines must begin with a hash sign (`#`). These directives are:

**#define** *name token-string*

Replaces subsequent instances of *name* with *token-string*.

**#define** *name(arg, . . . ,arg) token-string*

Replaces subsequent instances of the sequence *name (arg, . . . ,arg)* with *token-string*, where each occurrence of an *arg* in *token-string* is replaced by the corresponding token in the comma-separated list. Note that there must not be any space between *name* and the left parenthesis.

**#undef** *name*

Ignores the definition of *name* from this point on.

- #include "file"**  
**#include <file>** Includes at this point the contents of *file*, which **cpp** then processes.
- If you enclose *file* in double quotation marks (" "), **cpp** searches first in the directory of *infile*, second in directories named with the **-I** flag, and last in directories on a standard list .
- If you use the <*file*> notation, **cpp** searches for *file* only in the standard places. It does not search the directory in which *infile* resides.
- #line num ["file"]** Includes line control information for the next pass of the C compiler. *num* is the line number of the next line and *file* is the file from which it comes. If you omit "*file*", the current file name remains unchanged.
- #endif** Ends a section of lines begun by a test directive (**#if**, **#ifdef**, or **#ifndef**). Each test directive must have a matching **#endif**.
- #ifdef name** Places the subsequent lines in the output only if *name* has been defined by a previous **#define** and has not been undefined by an intervening **#undef**.
- #ifndef name** Places the subsequent lines in the output only if *name* has not been defined by a previous **#define** or has been undefined by an intervening **#undef**.
- #if expr** Places subsequent lines in the output only if *expr* evaluates to nonzero. All the binary nonassignment C operators, the **?:** operator, and the unary **-**, **!**, and **~** operators are legal in *expr*. The precedence of the operators is the same as that defined in the C Language. There is also a unary operator **defined**, which can be used in *expr* in these two forms:
- defined (name)**  
**defined name**
- This allows the utility of **#ifdef** and **#ifndef** in a **#if** directive. Only these operators, integer constants, and names which are known by **cpp** should be used in *expr*. The **sizeof** operator is not available.
- #else** Places subsequent lines in the output only if the expression in the preceding **#if** directive evaluates to False (and hence the lines following the **#if** and preceding the **#else** have been ignored).

You can nest the test directives and the possible **#else** directives.

---

## Flags

- C** Copies source file comments to the output file. If you omit this flag, **cpp** removes all comments (except those found on **cpp** directive lines).
- Dname[=def]** Defines *name* as in a **#define** directive. The default *def* is 1.
- Idir** Looks first in *dir*, then looks in the directories on the standard list for **#include** files with names that do not begin with a / (slash). See the previous discussion of **#include**.
- P** Preprocesses input without producing line control information for the next pass of the C compiler.
- Uname** Removes any initial definition of *name*, where *name* is a reserved symbol predefined by the preprocessor.

## Examples

1. To display the text that the preprocessor sends to the C compiler:

```
/lib/cpp pgm.c
```

This preprocesses `pgm.c` and displays the resulting text at the work station. You may want to see the preprocessor output when looking for errors in your macro definitions.

2. To create a file containing more readable preprocessed text:

```
/lib/cpp -P -C pgm.c pgm.i
```

This preprocesses `pgm.c` and stores the result in `pgm.i`. It omits line numbering information intended for the C compiler (**-P**), and includes program comments (**-C**).

3. To predefine macro identifiers:

```
/lib/cpp -DBUFFERSIZE=512 -DDEBUG pgm.c pgm.i
```

This defines `BUFFERSIZE` with the value 512 and `DEBUG` with the value 1 before preprocessing.

4. To use **#include** files located in nonstandard directories:

```
/lib/cpp -I/u/jim/include pgm.c
```

This looks in the current directory for quoted **#include** files, then in `/u/jim/include`, and then in the standard directories. It looks in `/u/jim/include` for angle-bracketed **#include** files (`< >`) and then in the standard directories.

## cpp

---

### Files

`/usr/include` Standard directory for `#include` files.

### Related Information

The following commands: “`cc`” on page 112 and “`m4`” on page 465.

---

## craps

---

### Purpose

Plays craps.

### Syntax

/usr/games/craps —

OL805188

### Description

The **craps** game plays a form of the game of craps that is played in Las Vegas. It simulates the **roller** while you place bets. Bet with the roller by making a positive bet or with the **House** by making a negative bet.

You start with a \$2000 bankroll. When the program prompts with `bet?`, you may bet all or part of your bankroll. If you bet more than your bankroll, the program repeats the prompt until you make a legal bet. Then the roller throws the dice. The payoff odds are one to one. The player wins depending on whether the bet is placed with the roller or with the House. The *first* roll is the roll immediately following a bet.

The following rules apply. On the first roll, 7 or 11 wins for the roller; 2, 3, or 12 wins for the House; and any other number becomes the **point** and you roll again (the next rule then applies). On subsequent rolls, the point wins for the roller; 7 wins for the House; and any other number rolls again.

If you lose your bankroll, the House prompts `marker?`, offering to lend you an additional \$2000. Accept the loan by responding `y` or `yes`. Any other response ends the game. When you hold markers, the House reminds you before a bet how many markers are outstanding. When you have markers and your bankroll exceeds \$2000, **craps** asks `Repay marker?` If you want to repay part or all of your loan, respond with `y` (or `yes`). If you have more than one marker, **craps** asks you `How many?` If you respond with a number greater than the number of markers you hold, it repeats the prompt until you enter a valid number. If you accumulate 10 markers (a total loan of \$20,000), **craps** tells you so and exits. If you accumulate a bankroll of more than \$50,000 while holding markers, the money owed is repaid automatically.

A bankroll of more than \$100,000 breaks the bank, and **craps** will prompt `New game?` To quit the game, press **INTERRUPT (Alt-Pause)**; **craps** displays whether you have won, lost, or broken even and exits.



# crash

---

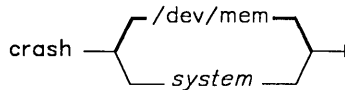
## crash

---

### Purpose

Examines system images.

### Syntax



OL805101

### Description

The **crash** command is an interactive utility for examining an operating system image (a core image or the running kernel). It has facilities for interpreting and formatting the various control structures in the system and certain miscellaneous functions useful for examining a dump.

The *system* parameter specifies the file that contains the system image and the kernel symbol definitions. Its default value is **/dev/mem**. Thus you can run **crash** with no arguments to examine an active system. If you specify a system-image file, **crash** assumes that it is a system dump file, and it sets the default process to the process running at the time of the crash.

**Note:** A source listing of the system header files may be helpful while using **crash** to identify the flags it uses.

Stack tracing of the current process on a running system does not work.

The **crash** command recognizes several aliases in the *format* specification accompanying the subcommands. They are as follows:

Format	Aliases	Format	Aliases
byte	b	inode	ino, i
character	char, c	longdec	ld, D
decimal	dec, e	longoct	lo, O
directory	direct, dir, d	octal	oct, o
hexadecimal	hexadec, hex, h, x	write	w

## Subcommands

The **crash** command presents a prompt ( $\triangleright$ ) when it is ready to interpret subcommands entered at the work station. The general subcommand format for **crash** is:

*subcommand* [*flags*] [*structures to be displayed*]

When allowed, *flags* modify the format of the data displayed. If you do not specify which structure elements you want to examine, all valid entries are displayed. In general, those subcommands that perform I/O with addresses assume hexadecimal notation.

Most of the subcommands recognized by **crash** have *aliases* (abbreviated forms that give the same result). **crash** recognizes the following subcommands:

- user** [*process-table-entry*] . . .           Aliases: **uarea, u\_area, u**  
 Displays the user structure of the named process as determined by the information contained in the process table entry. (See the `/usr/include/sys/user.h` file for this structure definition.) If you do not specify an entry, the information about the last running process is displayed. Attempting to display a paged process produces an error message.
- trace** [*process-table-entry*] . . .           Aliases: **t**  
 Displays a kernel stack trace of the current process. The trace starts at the bottom of the stack and attempts to find valid stack frames deeper in the stack. If you do not provide an entry number, information about the last running process is displayed.
- stack** [*process-table-entry*] . . .           Aliases: **stk, s, kernel, k**  
 Displays a dump of the kernel stack of a process. The addresses shown are virtual data addresses rather than true physical locations. If you do not provide an entry number, information about the last running process is displayed. Stack tracing of the current process on a running system does not work.
- proc** [-] [-r] [*process-table-entry*] . . .   Aliases: **ps, p**  
 Displays the process table. (See the `/usr/include/sys/proc.h` file for this structure definition.) The **-r** flag causes only runnable processes to be displayed. The **-** (minus) alone displays a longer listing.
- inode** [-] [*i-node-table-entry*] . . .       Aliases: **ino, i**  
 Displays the i-node table. The **-** flag also displays the i-node data block addresses. Unless specific i-node entries are requested, only those with a nonzero reference are displayed.
- file** [*file-table-entry*] . . .           Aliases: **files, f**  
 Displays the file table. Unless specific file entries are requested, only those with a nonzero reference are displayed.
- mount** [*mount-table-entry*] . . .       Aliases: **mnt, m**  
 Displays the mount table. Unless specific mount table entries are requested, only those in use are displayed.

# crash

---

- text** [*text-table-entry*] . . . Aliases: **txt, x**  
Displays the text table. Unless specific text entries are requested, only those with a nonzero i-node pointer are displayed.
- tty** [*type*] [-] [*tty-entry*] . . . Aliases: **term, dz, dh**  
Displays the tty structures. The *type* parameter specifies which structure is used (such as **ksr**, or **rs**). The last *type* entered with the **tty** command becomes the default. The - flag displays the **stty** parameters for the given line.
- stat** Displays statistics found in the dump. These include the panic message (if a panic occurred), time of crash, and system name.
- var** Aliases: **tunables, tunable, tune, v**  
Displays the tunable system parameters.
- buf** [*buffer-header*] . . .  
Displays the system buffer headers.
- buffer** [*format*] [*buffer*] . . .  
Displays the data in a system buffer according to *format*. If you do not provide a *format* parameter, the previous *format* is used. Valid formats include **decimal, octal, hex, character, byte, directory, i-node** and **write**. The **write** format creates a file in the current directory containing the buffer data.
- callout** Aliases: **calls, call, c, timeout, time, tout**  
Displays all entries in the callout table.
- map** [*map-name*] . . .  
Displays the named system map structures.
- nm** [*symbol*] . . .  
Displays symbol value and type as found in the *kernel-image* file.
- ts** [*text-address*] . . .  
Finds the text symbols closest to the given addresses.
- ds** [*data-address*] . . .  
Finds the data symbols closest to the given addresses.
- od** [*symbol name or address*] [*count*] [*format*]  
Dumps *count* data values starting at the symbol value or address given according to *format*. Allowable formats are octal, longoct, decimal, longdec, character, hex, or byte.
- !** Runs shell commands.
- q** Exits from **crash**.
- ?** Displays summary of **crash** commands.

## Files

/usr/include/sys/*.h	Header files for table and structure information.
/dev/mem	Default system-image file.
/unix	Default kernel-image file.
buf.#	Files containing buffer data.

## Related Information

The following commands: “**mount**” on page 498, “**nm**” on page 521, “**ps**” on page 579, “**sh**” on page 637, and “**stty**” on page 717.

## cron

---

## cron

---

### Purpose

Runs commands automatically.

### Syntax

cron —<sup>1</sup>

<sup>1</sup> Not usually run from the command line, but included in `/etc/rc`.

OL805184

### Description

The **cron** command runs shell commands at specified dates and times. Regularly scheduled commands can be specified according to instructions contained in **crontab** files. You can submit your **crontab** file via the **crontab** command (see page 174). Use the **at** command (see page 66) to submit commands that are to be run only once. Because **cron** never exits, it should be run only once. This is best done by running **cron** from the initialization process through the `/etc/rc` command file (see page 594).

The **cron** command examines **crontab** files and **at** command files only during process initialization and when a file changes. This reduces the overhead of checking for new or changed files at regularly scheduled intervals.

The **cron** command also executes a **sync** system call approximately once a minute to assure that all information in memory that should be on disk (buffered output) is written out. These periodic updates minimize the possibility of file system damage in the event of a crash. In addition, **cron** keeps a number of frequently used system directories open to keep their i-nodes in kernel memory for faster access.

If the file `/usr/lib/cron/log` exists, **cron** records a history of its activities in it.

For a discussion of how to schedule commands, see “**crontab**” on page 174.

### Files

<code>/usr/lib/cron</code>	Main cron directory.
<code>/usr/lib/cron/log</code>	Accounting information.
<code>/usr/spool/cron</code>	Spool area.
<code>/bin</code>	Directory kept open.
<code>/lib</code>	Directory kept open.

/usr	Directory kept open.
/usr/bin	Directory kept open.
/usr/lib	Directory kept open.
/etc	Directory kept open.
/tmp	Directory kept open.

## Related Information

The following commands: “**at**, **batch**” on page 66, “**crontab**” on page 174, and “**rc**” on page 594.

The **sync** system call and the **crontab** file in *AIX Operating System Technical Reference*.

# crontab

---

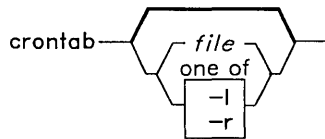
## crontab

---

### Purpose

Submits a schedule of commands to **cron**.

### Syntax



OL805003

### Description

The **crontab** command copies the specified *file*, or standard input if you do not specify a *file*, into a directory that holds all users' **crontab** files. The **cron** command runs commands according to the instructions in these **crontab** files. It then mails you the output from standard output and standard error for these commands, unless you redirect standard output or standard error. When entries are made to a **crontab** file, all previous entries are erased.

You may use **crontab** if your logname appears in the file `/usr/lib/cron/cron.allow`. If that file does not exist, **crontab** checks the file `/usr/lib/cron/cron.deny` to determine if you should be denied access to **crontab**. If neither file exists, you can submit a job only if you are operating with superuser authority. The allow/deny files contain one user name per line.

**Note:** If your login ID is associated with more than one login name, **crontab** uses the first login name that appears in the `/etc/passwd` file, regardless of which login name you might actually be using.

If **cron.allow** exists, the superuser's logname must appear there for the superuser to be able to use the command.

Each **crontab** file entry consists of a line with six fields, separated by spaces and tabs, that contain, respectively:

1. The minute (0-59)
2. The hour (0-23)
3. The day of the month (1-31)
4. The month of the year (1-12)
5. The day of the week (0-6 for Sunday-Saturday)
6. The shell command.

Each of these fields can contain:

- A number in the specified range
- Two numbers separated by a minus to indicate an inclusive range
- A list of numbers separated by commas, which selects all numbers in the list
- An asterisk, meaning all legal values.

Note that the specification of days may be made by two fields (day of the month and day of the week). If you specify both as a list of elements, both are adhered to. For example the following entry:

```
0 0 1,15 * 1 command
```

would run *command* on the first and fifteenth days of each month, as well as every Monday. To specify days by only one field, the other field should contain an **\***.

The **cron** command runs the command named in the sixth field at the selected date and time. If you include a **%** (percent sign) in the sixth field, **cron** treats everything that precedes it as the command invocation and makes all that follows it available to standard input, unless you escape or quote the percent sign (`\%` or `"%"`).

**Note:** The shell runs only the first line of the command field (up to a **%** or end of line). All other lines are made available to the command as standard input.

The **cron** command invokes a subshell from your **\$HOME** directory. This means that it will not run your **.profile** file. If you schedule a command to run when you are not logged in and you want to have commands in your **.profile** run, you must explicitly do so in the **crontab** file. (For a more detailed discussion of how **sh** can be invoked, see “**sh**” on page 637).

**cron** supplies a default environment for every shell, defining **HOME**, **LOGNAME**, **SHELL** (`=/bin/sh`), and **PATH** (`=:/bin:/usr/bin`).

## Flags

- l Lists your **crontab** file.
- r Removes your crontab file from the **crontab** directory.



## crontab

---

### Examples

The following examples show valid **crontab** file entries.

1. To write the time to the console every hour on the hour:

```
0 * * * * echo The hour is `date`. >/dev/console
```

This example uses **command substitution**. For more information, see “Command Substitution” on page 647.

2. To run **calendar** at 6:30 a.m. every Monday, Wednesday, and Friday:

```
30 6 * * 1,3,5 /usr/bin/calendar -
```

3. To define text for the standard input to a command:

```
0 16 10-31 12 5 wall%HAPPY HOLIDAYS!%Remember to turn in your time card.
```

This writes a message to all users logged in at 4:00 p.m. each Friday between December 10th and 31st.

The text following the % (percent sign) defines the standard input to the **wall** command as:

```
HAPPY HOLIDAYS!  
Remember to turn in your time card.
```

### Files

<code>/usr/lib/cron</code>	Main <b>cron</b> directory.
<code>/usr/spool/cron/crontabs</code>	Spool area.
<code>/usr/lib/cron/cron.allow</code>	List of allowed users.
<code>/usr/lib/cron/cron.deny</code>	List of denied users.

### Related Information

The following commands: “**cron**” on page 172 and “**sh**” on page 637.

---

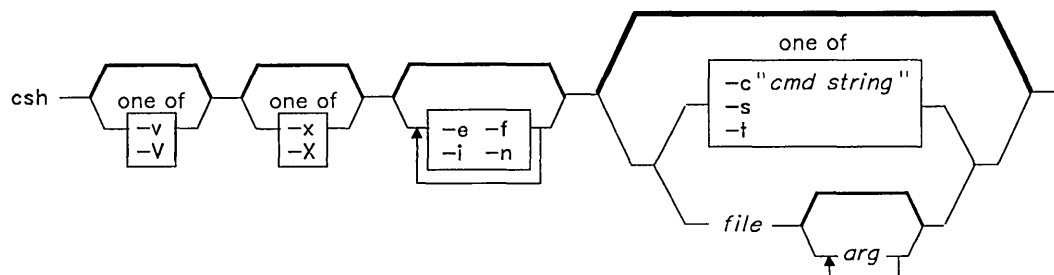
# csh

---

## Purpose

Interprets commands read from a file or entered from the keyboard.

## Syntax



OL805447

## Description

The **csh** command is a system command interpreter and programming language that incorporates a history mechanism and a C-like syntax. Like the **sh** command, it is an ordinary user program that reads commands typed at the keyboard and arranges for their execution. In addition, it can read commands from a file, usually called a *shell procedure* or a *command file*.

When you run **csh**, it begins by executing commands from the file **.cshrc** in your home directory, if it exists. If, on the other hand, **csh** runs as a login shell, it executes commands from your **.cshrc** file and your **.login** file.

## Commands

A *simple command* is a sequence of *words* separated by blanks or tabs. A word is a sequence of characters and/or numerals that does not contain unquoted blanks. In addition, the following characters and doubled characters also form single words when used as command separators or terminators:

```
& | ; < > ( )
&& !! << >>
```

These special characters may be parts of other words. Preceding them with a \ (backslash), however, prevents the shell from interpreting them as special characters. When the shell

is not reading input from a work station, it treats any word that begins with a # (number sign) as a comment and ignores that word and all characters following up to the next new-line character. Strings enclosed in matched pairs of quotation characters or grave accents ( ' ', " ", or ` `) can also form parts of words. (Blanks, tab characters, and special characters do not form separate words when they are found within these quotation marks.) In addition, within pairs of single quotation marks ( ' ') and double quotation marks ( " "), you may include the new-line character by preceding it with \ (backslash).

The first word in the simple-command sequence (numbered 0), usually specifies the name of a command. any remaining words, with a few exceptions, are passed to that command. If the command specifies an executable file that is a compiled program, the shell immediately runs that program. If the file is marked executable but is not a compiled program, the shell assumes that it is a shell procedure. In this case it spawns another instance of itself (a *subshell*), to read the file and execute the commands included in it.

A *pipeline* is a sequence of one or more commands separated by a | (vertical bar). The output of each command in a pipeline provides the input to the next command.

A *list* is a sequence of one or more pipelines separated by a ; (semicolon), & (ampersand), && (two ampersands), or || (two vertical bars) and optionally ended by a ; (semicolon) or an & (ampersand). These separators and terminators have the following effects:

- ; Causes *sequential execution* of the preceding pipeline (the shell waits for the pipeline to finish).
- & Causes *asynchronous execution* of the preceding pipeline (the shell does *not* wait for the pipeline to finish).
- && Causes the list following it to be executed *only* if the preceding pipeline returns a zero exit value.
- || Causes the list following it to be executed *only* if the preceding pipeline returns a nonzero exit value.

**Note:** The `cd` command is an exception. If it returns a nonzero exit value, no subsequent commands in a list are executed, regardless of the separators.

The ; and & separators have equal precedence, as do && and ||. The single-character separators have lower precedence than the double-character separators. A unquoted new-line character following a pipeline functions the same as a ; (semicolon).

Place any of the above in parentheses to form a simple command.

The shell associates a *job* with each pipeline. It keeps a table of current jobs and assigns them small integer numbers. When you start a job asynchronously by terminating the command with a &, the shell displays a line that looks like the following:

```
[1] 1234
```

This line indicates that the job number is 1 and that the job is composed of one process with a process-ID of 1234. Use the built-in **jobs** command (page 194) to see what jobs are currently running.

A job running in the background competes for input if it tries to read from the work station. Background jobs can also produce output that competes for the work station and is interleaved there with the output of other jobs.

There are several ways to refer to jobs in the shell. Use the % (percent) character to introduce a job name. This name can be either the job number or the command name that started the job, if this name is unique. So, for example, if a **make** process is running as job 1, you can refer to it as %1. You can also refer to it as %make, if there is only one suspended job with a name that begins with the string make. You can also use

*?:string*

to specify a job whose name contains *string*, if there is only one such job.

The shell detects immediately whenever a process changes state. Whenever a job becomes blocked so that further progress is not possible, a message is sent to the work station, but not until just before the shell prompt. If, however, the **notify** shell variable is set (see page 188), the shell issues a message that indicates changes in status of background jobs immediately. Use the **notify** built-in command (page 195) to mark a single process so that its status changes are immediately reported. By default, **notify** marks the current process.

## History Substitution

History substitution lets you use words from previous commands as portions of new commands, thus making it easy to repeat commands, repeat the arguments of a previous command in the current command, or fix spelling mistakes in the previous command with little typing.

History substitutions begin with the ! (exclamation) character and may appear anywhere on the command line, provided they do not nest (in other words, a history substitution cannot contain another history substitution). You can precede the ! with a \ to prevent its special meaning. In addition, if you place the ! before a blank, tab, new-line character, = (equal sign), or ( (left parenthesis), it is passed unchanged. History substitutions also occur when you begin an input line with a ^ (circumflex). (This special abbreviation is discussed on page 182.) The shell echoes any input line containing history substitutions at the work station before it executes that line.

The history list saves commands that the shell reads from the work station and that consist of one or more words. History substitution reintroduces sequences of words from these saved commands into the input stream.

The **history** shell variable (page 187) controls the size of the history list. You must set the **history** shell variable either in the **.cshrc** file or on the command line with the built-in **set** command (page 196). The previous command is always retained, however, regardless of

the value of **history**. Commands in the history list are numbered sequentially starting from 1.

The built-in **history** command (page 193) produces output of the type:

```
 9 write michael
10 ed write.c
11 cat oldwrite.c
12 diff *write.c
```

The command strings are shown with their event numbers. It is not usually necessary to use event numbers to refer to events, but you can have the current event number displayed as part of your system prompt by placing an **!** in the prompt string assigned to the **prompt** environmental variable (page 188).

A full history reference contains an event specification, a word designator, and one or more modifiers in the following general format:

*event[:word:modifier[:modifier]] . . .*

In the previous sample of **history** command output, the current event number is 13. Using this example, the following refer to previous events:

#### **Event Specification**

! <b>10</b>	Refers to event number 10
! <b>-2</b>	Refers to event number 11 (the current event minus 2)
! <b>d</b>	Refers to a command word beginning with <b>d</b> (in this case event number 12)
! <b>?mic?</b>	Refers to a command word that contains the string <b>mīc</b> (in this case, event number 9).

These forms, without further modification, simply reintroduce the words of the specified events, each separated by a single blank. As a special case, **!!** refers to the previous command; the command **!!** alone on an input line reruns the previous command.

To select words from an event, follow the event specification with a **:** (colon) and one of the following word designators (the words of an input line are numbered sequentially starting from 0):

#### **Word Designator**

<b>0</b>	The first word (the command name)
<b><i>n</i></b>	The <i>n</i> th argument
<b>^</b>	The first argument
<b>\$</b>	The last argument

---

%	The word matched by an immediately preceding <i>?string?</i> search
<i>x-y</i>	A range of words from the <i>x</i> th word to the <i>y</i> th word
- <i>y</i>	A range of words from the first word (0) to the <i>y</i> th word
*	The first through the last argument, or nothing if there is only one word (the command name) in the event
<i>x</i> *	The <i>x</i> th through the last argument
<i>x-</i>	Like <i>x</i> * but omitting the last word.

You may omit the colon that separates the event specification from the word designator if the word designator begins with a ^, \$, \*, -, or %. You can also place a sequence of the following modifiers after the optional word designator, each preceded by a colon:

**Modifier**

<b>h</b>	Remove a trailing path name extension, leaving the head.
<b>r</b>	Remove a trailing “.xxx” component, leaving the root name.
<b>e</b>	Remove all but the trailing extension “.xxx.”
<b>s/l/r/</b>	Substitute <i>l</i> for <i>r</i> . With substitutions, it is an error for no word to be applicable.

The left side of a substitution is not a patterns in the sense of the editors but, rather, a string. Normally, a / (slash) delimits the string (*l*) and its replacement (*r*). However, you can use any character as the delimiter if you precede that character with a \ (backslash). Thus, in the following example:

```
s\%/usr/myfile\%/usr/yourfile%
```

the % becomes the delimiter allowing you to include the / in your strings. If you include an & in the replacement string, it is replaced by the text from the left-hand side (*l*). A null *l* string is replaced by either the last substitution or by the last string used in the contextual scan *!?string?*.

You may omit the trailing delimiter (/) if a new-line character follows immediately.

<b>t</b>	Remove all leading path name components, leaving the tail.
<b>&amp;</b>	Repeat the previous substitution.
<b>g</b>	Apply the change globally, that is, <b>g&amp;</b> .
<b>p</b>	Display the new command, but do not run it.
<b>q</b>	Quote the substituted words, thus preventing further substitutions.
<b>x</b>	Act like <b>q</b> , but break into words at blanks, tabs, and new-line characters.

Unless the modifier is preceded by a **g**, the change applies only to the first modifiable word.

If you give a history reference without an event specification, for example. `!$`, the shell uses the previous command as the event, unless a previous history reference occurs on the same line, in which case it repeats the previous reference. Thus, the following sequence:

```
!?foo?^ !$
```

gives the first and last arguments of the command that matches `?foo?`

A special abbreviation of a history reference occurs when the first nonblank character of an input line is a `^` (circumflex). This is equivalent to `!:s^`, thus providing a convenient shorthand for substitutions on the text of the previous line. The command `^|b^|ib` corrects the spelling of `lib` in the previous command.

You can enclose a history substitution in `{}` (braces), if necessary, to insulate it from the characters that follow. For example, if you want to use a reference to the command:

```
ls -ld ~paul
```

to perform the command:

```
ls -ld ~paula
```

use the following:

```
!{l}a
```

whereas `!|a` would look for a command starting with `|a`

### Quoting with Single and Double Quotes

Enclose strings in single and double quotation marks to prevent all or some of the substitutions that remain. Enclosing strings in single quotation marks (`' '`) prevents any further interpretation. Enclosing strings in double quotation marks (`" "`) allows further expansion. In both cases, the text that results becomes (all or part of) a single word. Only in one special case does a string quoted by `" "` yield parts of more than one word; strings quoted by `' '` never do (see “Command Substitution” on page 183).

### Command and File Name Substitution

The shell performs command and file-name substitutions selectively on the arguments of built-in commands. This means that it does not expand those parts of expressions that are not evaluated. For nonbuilt-in commands, the shell substitutes the command name separately from the argument list. This occurs very late, after it performs input/output redirection and in a child of the main shell.

## *Command Substitution*

The shell performs command substitution on a command string enclosed in grave accents ( ` ` ). The shell normally breaks the output from such a command into separate words at blanks, tabs, and new-line characters; this text then replaces the original command string. Within strings surrounded by double quotation marks ( " " ), the shell treats only the new-line character as a word separator, thus preserving blanks and tabs within the word.

In any case, the single final new-line character does not force a new word. Note that it is therefore possible for command substitution to yield only part of a word, even if the command outputs a complete line.

## *File-name Substitution*

If a word contains any of the characters \*, ?, [, or {, or begins with the ~ character, then that word is a candidate for file name substitution, also known as *globbing*. This word is then regarded as a pattern and replaced with an alphabetically sorted list of file names which match the pattern. The current collating sequence is used, which may be specified by the environment variables NLCTAB or NLFILÉ. In a list of words specifying file name substitution, it is an error for no patterns to match an existing file name, but it is not required that each pattern match. Only the character-matching symbols \*, ?, and [ imply pattern matching; the characters ~ and { being more related to abbreviations.

In matching file names, the character . (dot) at the beginning of a file name or immediately following a /, and the character /, must be matched explicitly. The \* character matches any string of characters, including the null string. The ? character matches any single character. The sequence [abcd] matches any one of the enclosed characters. Within [], a lexical range of characters may be indicated by [a-z]. The characters that match this pattern are defined by the current collating sequence (see “ctab” on page 204).

The ~ character at the beginning of a file name is used to see home directories. Standing alone, ~ expands to your home directory as reflected in the value of the home shell variable. When followed by a name that consists of letters, digits, and - characters, the shell searches for a user with that name and substitutes their home directory. Thus, ~ken might expand to /usr/ken and ~ken/chmach to /usr/ken/chmach. If the ~ character is followed by a character other than a letter or /, or appears not at the beginning of a word, it is left undisturbed.

The pattern a{b,c,d}e is a shorthand for abe ace ade. The shell preserves the left-to-right order, with results of matches being stored separately at a low level to preserve this order. This construct may be nested. Thus:

```
~source/sl/{oldls,ls}.c
```



expands to:

```
/usr/source/sl/oldls.c /usr/source/sl/ls.c
```

if the home directory for source is /usr/source. Similarly:

```
../{memo,*box}
```

might expand to:

```
../memo ../box ../mbox
```

(Note that memo is not sorted with the results of matching \*box.) As a special case, {, }, and {} are passed undisturbed.

### Alias Substitution

The shell maintains a list of aliases that the **alias** and **unalias** built-in commands (page 190) can establish, display, and modify. After the shell scans the command line, it divides it into distinct commands and checks the first word of each command, left to right, to see if it has an alias. If it does, the shell uses the history mechanism available (see “History Substitution” on page 179), to replace the text of the alias with the text of the command it stands for. The words that result replace the command and argument list. If reference is not made to the history list, then the argument list is left unchanged. Thus, if the alias for the **ls** command is `ls -l`, the shell replaces the command `ls /usr` with `ls -l /usr`, the argument list here being undisturbed because there is no reference to the history list in aliased command. Similarly, if the alias for **lookup** is:

```
grep !^ /etc/passwd
```

then the shell replaces `lookup bill` with:

```
grep bill /etc/passwd
```

Here, `!^` refers to the history list and the shell replaces it with the first argument in the input line, in this case `bill`.

Note from this last example that you can use special pattern-matching characters in an alias. Thus the command:

```
alias lprint 'pr !* >> print'
```

makes a command which formats its arguments to the line printer. The `!` is protected from the shell in the alias so that it is not expanded until `pr` runs.

If an alias is found, the word transformation of the input text is performed and the aliasing process begins again on the reformed input line. If the first word of the next text is the same as the old, looping is prevented by flagging it to prevent further aliasing. Other loops are detected and cause an error.

## Variable Substitution

The shell maintains a set of variables, each of which has as its value a list of zero or more words. Some of these variables are set by the shell or referred to by it. For instance, the **argv** variable is an image of the shell variable list, and words which comprise the value of this variable are referred to in special ways.

You can change and display the values of variables with the **set** and **unset** commands. Of the variables referred to by the shell, a number are toggles; the shell does not care what their value is, only whether they are set or unset. For instance, the **verbose** variable is a toggle which causes command input to be echoed. The setting of this variable results from the **-v** flag on the command line.

Other operations treat variables numerically. The **@** command performs numeric calculations and the result is assigned to a variable. Variable values are, however, always represented as (zero or more) strings. For numeric operations, the null string is considered to be zero, and the second and subsequent words of multi-word values are ignored.

After an input line is aliased and parsed, and before each command is run, variable substitution is performed, keyed by **\$** characters. You can prevent this expansion by preceding the **\$** with a **\**, except within **" "** (double quotation marks, where it always occurs, and within **' '** (single quotation marks), where it never occurs. Strings quoted by **' '** are interpreted later (see “Command Substitution” on page 183), so **\$** substitution does not occur there until later, if at all. A **\$** is passed unchanged if it is followed by a blank, tab, or new-line character.

Input/output redirections are recognized before variable expansion and are variable expanded separately. Otherwise, the command name and complete argument list expands together. It is therefore possible for the first (command) word to this point to generate more than one word, the first of which becomes the command name and the rest of which become parameters.

Unless enclosed in **" "** or given the **:q** modifier, the results of variable substitution may themselves eventually be command and file name substituted. Within pairs of double quotation marks, a variable with a value that consists of multiple words expands to a (portion of a) single word, with the words of the variable’s value separated by blanks. When you apply the **:q** modifier to a substitution, the variable expands to multiple words. Each word is separated by a blank and quoted to prevent later command or file name substitution.

The following notation allows you to introduce variable values into the shell input. Except as noted, it is an error to reference a variable that is not set.

*?name*  
*?{name}*

Replaced by the words assigned to *name*, each separated by a blank. Braces insulate *name* from any following characters that would otherwise be part of it. Shell variable names start with a letter and consist of up to 20 letters and digits, including the   (underline) character. If *name* is not a shell variable but is set

in the environment, then that value is returned. The `:` modifiers and the other forms given below are not available in this case.

`?name[selector]`  
`?{name[selector]}`

Used to select only some of the words from the value of *name*. The selector is subjected to `$` substitution and may consist of a single number, or two numbers separated by a `-`. The first word of a variable's string value is numbered 1. If the first number of a range is omitted, it defaults to 1. If the last member of a range is omitted, it defaults to  `$#name`. The `*` symbol selects all words. It is not an error for a range to be empty if the second argument is omitted or is in range.

`?#name`  
`?{#name}`

Gives the number of words in the variable. This is useful for later use in a `[selector]`.

`?0`

Substitutes the name of the file from which command input is being read. An error occurs if the name is not known.

`?number`  
`?{number}`

Equivalent to `$argv[number]`

`?*`

Equivalent to `$argv[*]`.

You can apply the modifiers `:h`, `:r`, `:q`, and `:x` to the substitutions above, as may `:gh`, `:gt` and `:gr`. If `{}` (braces) appear in the command form, then the modifiers must appear within the braces. The current implementation allows only one `:` modifier on each `$` expansion.

The following substitutions may not be changed with `:` modifiers.

`$?name`  
 `${?name}`

Substitutes the string 1 if name is set; 0 if it is not set.

`$?0`

Substitutes 1 if the current input file name is known; 0 if it is not known.

`$$`

Substitutes the (decimal) process number of the (parent) shell.

`$<`

Substitutes a line from the standard input, without further interpretation. Use it to read from the keyboard in a shell procedure.

---

## Predefined and Environmental Variables

The following variables have special meaning to the shell. Of these, **argv**, **cwd**, **home**, **path**, **prompt**, **shell**, and **status** are always set by the shell. Except for **cwd** and **status**, this setting occurs only at initialization. These variables are not changed unless this is done explicitly by you.

The **csh** command copies the environment variables **USER**, **TERM**, **HOME**, and **PATH** into the **csh** variables **user**, **term**, **home**, and **path**, respectively. The values are copied back into the environment whenever the normal shell variables reset. It is not necessary to worry about the setting of the **path** variable other than in the **.cshrc** file, since **csh** subprocesses import the definition of **path** from the environment and re-export it if it is changed.

<b>argv</b>	Set to the arguments to the shell; it is from this variable that positional parameters are substituted.
<b>cdpath</b>	Can be given a list of alternate directories to be searched by the <b>chdir</b> commands to find subdirectories.
<b>cwd</b>	The full path name of the current directory.
<b>echo</b>	Set when the <b>-x</b> command line flag is used; when set, causes each command and its arguments to echo just before it is run. For non built-in commands, all expansions occur before echoing. Built-in commands are echoed before command and file name substitution, since these substitutions are then done selectively.
<b>histchars</b>	Can be given a string value to change the characters used in history substitution. Use the first character of its value as the history substitution character, this replaces the default character <b>!</b> . The second character of its value replaces the <b>^</b> (circumflex) character in quick substitutions.
<b>history</b>	Can be given a numeric value to control the size of the history list. Any command that is referenced in this many events is not discarded. Very large values of <b>history</b> may run the shell out of memory. Saves the last command that ran on the history list, regardless of whether <b>history</b> is set.
<b>home</b>	Your home directory, initialized from the environment. The file name expansion of <b>~</b> refers to this variable.
<b>ignoreeof</b>	If set, the shell ignores an end-of-file character from input devices that are work stations. This prevents shells from accidentally being killed when it reads an end-of-file character ( <b>Ctrl-D</b> ).
<b>mail</b>	The files where the shell checks for mail. This is done after each command completion, which results in a prompt if a specified interval has elapsed. The shell displays the message, "You have new mail" if the file exists with an access time not greater than its change time.

	<p>If the first word of the value of <b>mail</b> is numeric, it specifies a different mail checking interval (in seconds); the default is 10 minutes.</p> <p>If you specify multiple mail files, the shell displays the message, "New mail in <i>file</i>", when there is mail in <i>file</i>.</p>
<b>noclobber</b>	<p>If set, places restrictions on output redirection to insure that files are not accidentally destroyed, and that &gt; &gt; redirections see existing files. (See "Redirecting Input and Output" on page 189)</p>
<b>noglob</b>	<p>If set, inhibits file name expansion. This is most useful in shell procedures that are not dealing with file names, or after a list of file names has been obtained and further expansions are not desirable.</p>
<b>nonomatch</b>	<p>If set, it is not an error for a file name expansion to not match any existing files; rather, the primitive pattern returns. It is still an error for the primitive pattern to be malformed.</p>
<b>notify</b>	<p>If set, the shell notifies asynchronously of changes in job status. The default presents status changes just before displaying the shell prompt.</p>
<b>path</b>	<p>Each word of the <b>path</b> variable specifies a directory in which commands are to be sought for execution. A null word specifies the current directory. If there is no <b>path</b> variable set, then only full path names run. The usual search path is the current directory, <b>/bin</b>, and <b>/usr/bin</b>. For the superuser, the default search path is <b>/etc</b>, <b>/bin</b>, and <b>/usr/bin</b>. A shell which is given neither the <b>-c</b> nor the <b>-t</b> flags normally hashes the contents of the directories in the <b>path</b> variable after reading <b>.cshrc</b> and each time the <b>path</b> variable is reset. If new commands are added to these directories while the shell is active, it may be necessary to give the <b>rehash</b> command (page 196), or the commands may not be found.</p>
<b>prompt</b>	<p>The string which is displayed before each command is read from an interactive work station input. If a <b>!</b> appears in the string, it is replaced by the current even number, unless a preceding <b>\</b> is given. The default prompt is <b>%</b>, <b>#</b> for the superuser.</p>
<b>savehist</b>	<p>Given a numeric value to control the number of entries of the history list that are saved in <b>~/.history</b> when you log out. Any command which is referenced in this many events is saved. During startup, the shell reads <b>~/.history</b> into the history list, enabling history to be saved across logins. Very large values of <b>savehist</b> slow down the shell startup.</p>
<b>shell</b>	<p>The file in which the shell resides. This is used in forking shells to interpret files which have execute bits set, but which are not executable by the system (see "Nonbuilt-in Command Execution" on page 199). This is initialized to the (system-dependent) home of the shell.</p>

<b>status</b>	The status returned by the last command. If it ended abnormally, then 0200 is added to the status. Built-in commands that fail return exit status 1; all other built-in commands set status 0.
<b>time</b>	Controls automatic timing of commands. If set, then any command that takes more than this many CPU seconds cause a line giving user, system, and real times and a utilization percentage, that is the ratio of user-plus-system-times to real time, displays when it ends.
<b>verbose</b>	Set by the <b>-v</b> command line flag, causes the words of each command to display after history substitution.

## Redirecting Input and Output

You can redirect the standard input and standard output of a command with the following syntax:

< <i>name</i>	Opens file <i>name</i> (which is first variable, command, and file name expanded) as the standard input.
<< <i>word</i>	Reads the shell input up to a line which is the same as <i>word</i> . <i>word</i> is not subjected to variable, file name, or command substitution, and each input line is compared to <i>word</i> before any substitutions are done on this input line. Unless a quoting character (\, ", ', or `) appears in <i>word</i> , the shell performs variable and command substitution on the intervening lines, allowing \ to quote \$, \, and `. Commands which are substituted have all blanks, tabs, and new-line characters preserved, except for the final new-line character, which is dropped. The resultant text is placed in an anonymous temporary file, which is given to the command as standard input.
> <i>name</i>	
>! <i>name</i>	
>& <i>name</i>	
>&! <i>name</i>	Uses the file <i>name</i> as standard output. If the file does not exist, it is made. If the file exists, it is truncated, its previous contents being lost.  If the <b>noclobber</b> shell variable is set, the file must not exist or be a character special file, or an error results. This helps prevent accidental destruction of files. In this case, use the <b>!</b> forms to suppress this check.  The forms involving <b>&amp;</b> route the diagnostic output into the specified file as well as the standard output. <i>name</i> expands in the same way as < input file names.

>> *name*  
 >>& *name*  
 >>! *name*  
 >>&! *name* Uses file *name* as standard output like >, but places output at the end of the file. If the **noclobber** shell variable is set, it is an error for the file not to exist, unless one of the ! forms is given. Otherwise, it is similar to >.

A command receives the environment in which the shell was invoked, as changed by the input/output parameters and the presence of the command as a pipeline. Thus, unlike some previous shells, commands that run from a file of shell commands do not have any access to the text of the commands by default. Rather, they receive the original standard input of the shell. Use the << mechanism to present inline data. This lets shell command files function as components of pipelines and lets the shell block read its input. Note that the default standard input for a command run detached is not changed to be the empty file /dev/null. Rather, the standard input remains as the original standard input of the shell.

To redirect the diagnostics output through a pipe with the standard output, use the form !& rather than just | (vertical bar).

## Control Flow

The shell contains some commands that can be used to regulate the flow of control in command files (shell procedures) and (in limited but useful ways) from work station input. These commands all operate by forcing the shell to reread or skip in its input and, because of the implementation, restrict the placement of some of the commands.

The **foreach**, **switch**, and **while** statements, and the **if-then-else** form of the **if** statement, require that the major keywords appear in a single simple command on an input line.

If the shell input is not searchable, the shell buffers input whenever a loop is being read and searches the internal buffer to do the rereading implied by the loop. To the extent that this allows, backward **gotos** succeed on inputs that you cannot search.

## Built-in Commands

Built-in commands are run within the shell. If a built-in command occurs as any component of a pipeline except the last, it runs in a subshell.

**alias**

**alias** *name*

**alias** *name wordlist*

Displays all aliases (first form). The second form displays the alias for *name*. The final form assigns the specified *wordlist* as the alias of *name*. *wordlist* is command and file name substituted. *name* is not allowed to be **alias** or **unalias**.

---

<b>break</b>	Resumes running after the end of the nearest enclosing <b>foreach</b> or <b>while</b> . Runs the remaining commands on the current line. Multi-level breaks are therefore possible by writing them all on one line.
<b>breaksw</b>	Breaks from a <b>switch</b> ; resumes after the <b>endsw</b> .
<b>case label:</b>	Defines a label in a <b>switch</b> statement, as discussed in the following.
<b>cd</b> <b>cd name</b> <b>chdir</b> <b>chdir name</b>	Changes the current directory to <i>name</i> . If no argument is given, then changes to your home directory.  If <i>name</i> is not found as a subdirectory of the current directory and does not begin with <i>/</i> , <i>./</i> , or <i>../</i> , then each component of the <b>cdpath</b> shell variable is checked to see if it has a subdirectory <i>name</i> . Finally, if all else fails, but <i>name</i> is a shell variable with a value that begins with <i>/</i> , then this is tried to see if it is a directory.
<b>continue</b>	Continues execution of the nearest enclosing <b>while</b> or <b>foreach</b> . The rest of the commands on the current line run.
<b>default:</b>	Labels the default case in a <b>switch</b> statement. The default should come after all <b>case</b> labels.
<b>dirs</b>	Displays the directory stack, the top of the stack is at the left, the first directory in the stack being the current directory.
<b>echo string</b> <b>echo -n string . . .</b>	Writes the listed <i>strings</i> to the shell's standard output, separated by spaces and ending with a new-line character unless you specify the <b>-n</b> flag.
<b>else</b> <b>end</b> <b>endif</b> <b>endsw</b>	See the description of the <b>foreach</b> , <b>if</b> , <b>switch</b> , and <b>while</b> statements.
<b>eval arg . . .</b>	Reads <i>arg</i> as input to the shell and runs the resulting command(s) in the context of the current shell. Use this to run commands generated as the result of command or variable substitution, since parsing occurs before these substitutions.
<b>exec cmd</b>	Runs the specified command in place of the current shell.



<b>exit</b>	
<b>exit</b> ( <i>expr</i> )	Exits the shell with either the value of the <b>status</b> shell variable (first form) or with the value of the specified expression (second form).
<b>foreach</b> <i>name</i> ( <i>list</i> )	
...	
<b>end</b>	Successively sets <i>name</i> to each member of <i>list</i> and runs the sequence of commands between the <b>foreach</b> and the matching <b>end</b> . Both <b>foreach</b> and <b>end</b> must appear alone on separate lines.
	Use the <b>continue</b> statement to continue the loop and the <b>break</b> statement to end the loop prematurely. When this command is read from the work station, the loop is read once, prompts with ? before any statement in the loop runs. If a mistake is made in entering a loop, it can be corrected before you run the loop. Commands within loops, prompted for by ?, are not placed in the history list.
<b>glob</b> <i>list</i>	Functions like <b>echo</b> , but does not recognize backslash (\) escapes, and delimits words by null characters in the output. Useful for programs that wish to use the shell to file name expand a list of words.
<b>goto</b> <i>word</i>	Continues to run after the line specified by <i>word</i> . The specified <i>word</i> is file-name and command expanded to yield a string of the form <i>label</i> . The shell rewinds its input as much as possible and searches for a line of the form <i>label</i> :, possibly preceded by blanks or tabs.
<b>history</b>	
<b>history</b> <i>num</i>	
<b>history -r</b> <i>num</i>	
<b>history -h</b> <i>num</i>	Displays the history event list. If you specify a number, only the <i>n</i> most recent events are displayed. The <b>-r</b> flag reverses the order of display to the most recent first rather than the oldest first. The <b>-h</b> flag causes the history list to be displayed without leading numbers. Use this to produce files suitable for used with the <b>-h</b> flag of the <b>source</b> command.
<b>if</b> ( <i>expr</i> ) <i>cmd</i>	Runs the single command (with arguments) if the specified expression evaluates true. Variable substitution on <i>cmd</i> happens early, at the same time it does for the rest of the <b>if</b> statement. <i>cmd</i> must be a simple command, not a pipeline, command list, or parenthesized command list.
	<b>Note:</b> Input and output redirection occurs even if <i>expr</i> is false (and the command is not executed).

```

if (expr) then
...
else if (expr2) then
...
else
...
endif

```

If *expr* is true, runs the commands that follow the first **then**; **else if** *expr2* is true, runs the commands that follow the second **then**; **else** runs the commands that follow the second **else**. Any number of **else-if** pairs are possible; only one **endif** is needed. The **else** part is optional. The words **else** and **endif** must appear at the beginning of input lines. The **if** must appear alone on its input line or after an **else**.

```

jobs
jobs -l

```

Lists the active jobs. With the **-l** flag, lists process-IDs in addition to the job number and process-ID.

```

kill %job
kill -signal %job ...
kill pid
kill -signal pid ...
kill -l

```

Sends to the jobs or process that you specify either the **TERM** (terminate) signal or *signal*. Specify *signals* either by number or by names (as given in `/usr/include/signal.h`, stripped of the SIG prefix). Signal names are listed by **kill -l**.

```

limit
limit resource
limit resource max-use

```

Limits the usage by the current process and each process it creates to not individually exceed *max-use* on the specified *resource*. If a *max-use* is not given, the current limit displays; if a *resource* is not given, all limitations are given.

Controllable resources are limited to **filesize**, **stacksize**, and **datasize**.

You can specify *max-use* as a (floating-point or integer) number followed by a scale factor: **k** or **kilobytes** (1024 bytes), **m** or **megabytes**, or **b** or **blocks** (the units used by the **ulimit** system call).

For both *resource* names and scale factors, unambiguous prefixes of the names suffice.

**filesize** may be lowered by an instance of **csh**, but may only be raised by an instance whose effective user-ID is `root`. (See the **ulimit** system call in *AIX Operating System Technical Reference*.)

<b>login</b>	Ends a login shell, and replaces it with an instance of <b>/bin/login</b> . This is one way to log out (included for compatibility with the <b>sh</b> command).
<b>logout</b>	Ends a login shell. Especially useful if <b>ignoreeof</b> is set.
<b>nice</b> <b>nice + num</b> <b>nice cmd</b> <b>nice + num cmd</b>	Sets the priority of commands run in this shell to 24 (first form). The second form sets the priority to the specified number. The final two forms run the specified command at priority 24 and the specified number, respectively. If you are have superuser authority you can specify <b>nice</b> with a negative number. The command always runs in a subshell, and the restrictions placed on commands in simple <b>if</b> statements apply.
<b>nohup</b> <b>nohup cmd</b>	Causes hangups to be ignored for the remainder of the procedure (first form). The second form causes the specified command to be run with hangups ignored. All processes run in the background with <b>&amp;</b> are effectively protected from being sent a hangup signal when you log out, but will still be subject to explicitly sent hangups unless <b>nohup</b> is used.
<b>notify</b> <b>notify %job ...</b>	Causes the shell to notify you asynchronously when the status of the current or specified jobs changes. Normally, notification is presented just before the shell prompt. This is automatic if the <b>notify</b> shell variable is set.
<b>onintr</b> <b>onintr -</b> <b>onintr label</b>	Controls the action of the shell on interrupts. The first form restores the default action of the shell on interrupts, which is to end shell procedures or to return to the work station command input level. The second form causes all interrupts to be ignored. The third form causes the shell to run a <b>goto label</b> when it receives an interrupt or a child process ends due to an interruption.  In any case, if the shell is running detached and interrupts are being ignored, all forms of <b>onintr</b> have no meaning, and interrupts continue to be ignored by the shell and all invoked commands.
<b>popd</b> <b>popd + n</b>	Pops the directory stack, returns to the new top directory. With a <b>+ n</b> , discards the <i>n</i> th entry in the stack. The elements of the directory stack are numbered from the top starting at 0.

---

<p><b>pushd</b>  <b>pushd</b> <i>name</i>  <b>pushd</b> <i>+n</i></p>	<p>With no arguments, exchanges the top two elements of the directory stack. With <i>name</i>, changes to the new directory and pushes the old current directory (as given in the <b>cwd</b> shell variable) onto the directory stack. With a numeric argument, rotates the <i>n</i>th argument of the directory stack around to be the top element and changes to it. The members of the directory stack are numbered from the top starting at 0.</p>
<p><b>rehash</b></p>	<p>Causes the internal hash table of the contents of the directories in the <b>path</b> shell variable to be recomputed. This is needed if new commands are added to directories in <b>path</b> while you are logged in. This should only be necessary if commands are added to one of the user's own directories, or if someone changes the contents of one of the system directories.</p>
<p><b>repeat</b> <i>count cmd</i></p>	<p>Runs the specified command, which is subject to the same restrictions as the <b>if</b> statement, <i>count</i> times.</p> <p><b>Note:</b> I/O redirections occur exactly once, even if <i>count</i> is 0.</p>
<p><b>set</b>  <b>set</b> <i>name</i>  <b>set</b> <i>name = word</i>  <b>set</b> <i>name [index] = word</i>  <b>set</b> <i>name = (list)</i></p>	<p>Shows the value of all shell variables (first form). Variables that have more than a single word as their value are displayed as a parenthesized word list. The second form sets <i>name</i> to the null string. The third form sets the <i>index</i>th component of <i>name</i> to <i>word</i>; this component must already exist. The final form sets <i>name</i> to the list of words in <i>list</i>. In all cases, the value is command- and file-name expanded.</p> <p>These arguments may be repeated to set multiple values in a single <b>set</b> command. However, variable expansion happens for all arguments before any setting occurs.</p>
<p><b>setenv</b> <i>name value</i></p>	<p>Sets the value of environment variable <i>name</i> to be <i>value</i>, a single string. The most commonly used environment variables, <b>USER</b>, <b>TERM</b>, and <b>PATH</b>, are automatically imported to and exported from the <b>csh</b> variables <b>user</b>, <b>term</b>, and <b>path</b>; there is no need to use <b>setenv</b> for these.</p> <p>If you modify the environment variables <b>NLFILE</b> or <b>NLCTAB</b>, the current international character support environment and collating sequence are changed as specified for subsequent commands executed from the shell.</p>

**shift**  
**shift** *variable*

Shifts the members of **argv** to the left. It is an error for **argv** not to be set or to have less than one word as its value. The second form does the same function on the specified *variable*.

**source** *name*  
**source -h** *name*

Reads commands from *name*. You can nest the **source** commands. However, if they are nested too deeply, the shell may run out of file descriptors. An error in a **source** command at any level ends all nested **source** commands. Normally, input during **source** commands is not placed on the history list. The **-h** flag causes the commands to be placed in the history list without running.

**switch** (*string*)  
**case** *str1*:  
 ...  
**breaksw**  
**default**:  
 ...  
**breaksw**  
**endsw**

Successively matches each case label against *string*. The *string* is command and file-name expanded first. Use the pattern-matching characters **\***, **?**, and **[ . . . ]** in the case labels, which are variable expanded. If none of the labels match before a **default** label is found, then the execution begins after the **default** label. Each **case** label and the **default** label must appear at the beginning of a line. The **breaksw** command causes execution to continue after the **endsw**. Otherwise, control may fall through case labels and the **default** labels, as in C. If no label matches and there is no **default**, execution continues after the **endsw**.

**time**  
**time** *cmd*

With no argument, displays a summary of time used by this shell and its children. If arguments are given, the specified command is timed, and a time summary as described under the **time** shell variable is displayed. If necessary, an extra shell is created to display the time statistic when the command completes.

**umask**  
**umask** *value*

Displays the file creation mask (first form) or sets it to the specified *value* (second form). The mask is given as an octal value. Common values for the mask are 002, giving all access to owner and group and read and execute access to others, or 022, giving all access to the owner and all access except write access for users in the group or others.

---

<b>unalias</b> <i>pattern</i>	Discards all aliases with names that match <i>pattern</i> . Thus, all aliases are removed by <b>unalias *</b> . It is not an error for nothing to be unaliased.
<b>unhash</b>	Disables the use of the internal hash table to locate running programs.
<b>unlimit</b> <b>unlimit</b> <i>resource</i>	Removes the limitation on <i>resource</i> . If you do not specify <i>resource</i> , then all <i>resource</i> limitations are removed. The only removable limitation is that on <b>filesize</b> , and only the superuser can remove it.
<b>unset</b> <i>pattern</i>	Removes all variables with names that match the <i>pattern</i> . Use <b>unset *</b> to remove all variables. It is not an error for nothing to be unset.
<b>unsetenv</b> <i>pattern</i>	Removes all variables from the environment whose names match the specified <i>pattern</i> . (See the <b>setenv</b> built-in command on page 195.)
<b>wait</b>	Waits for all background jobs. If the shell is interactive, an INTERRUPT ( <b>Alt-Pause</b> ) can disrupt the wait, when the shell displays the names and job numbers of all jobs known to be outstanding.
<b>while</b> ( <i>expr</i> ) ... <b>end</b>	Evaluates the commands between the <b>while</b> and the matching <b>end</b> while <i>expr</i> evaluates nonzero. You can use <b>break</b> to end and <b>continue</b> to continue the loop prematurely. The <b>while</b> and <b>end</b> must appear alone on their input lines. If the input is a work station, prompts occur the first time through the loop, as for the <b>foreach</b> statement.
<b>@</b> <b>@</b> <i>name</i> = <i>expr</i> <b>@</b> <i>name</i> [ <i>index</i> ] = <i>expr</i>	Displays the values of all the shell variables (first form). The second form sets the specified <i>name</i> to the value of <i>expr</i> . If the expression contains <, >, &, or !, then at least this part of the expression must be placed within parentheses. The third form assigns the value of <i>expr</i> to the <i>indexth</i> argument of <i>name</i> . Both <i>name</i> and its <i>indexth</i> component must already exist.
	C operators, such as *= and += are available. The space separating <i>name</i> from the assignment operator is optional. Spaces are, however, required in separating components of <i>expr</i> , which would otherwise be single words. Special postfix ++ and -- operators increase and decrease <i>name</i> .

## Expressions

The `@` built-in command and the `exit`, `if`, and `while` statements accept expressions which include operators similar to those of C, with the same precedence. The following operators are available:

```
*   /   %
+   -
<< >>
<= >= >
==  !=  =~  !~
```

In the preceding list, operators of equal precedence appear on the same line, below those lines containing operators (if any) that have greater precedence and above those lines containing operators having lesser precedence. The `==`, `!=`, `=~`, and `!~` operators compare their arguments as strings; all others operate on numbers. The `=~` and `!~` operators are similar to `!=` and `==`, except that the right-most side is a *pattern* against which the left-hand operand is matched. This reduces the need for use of the `switch` statement in shell procedures when all that is really needed is pattern matching.

Strings which begin with 0 are considered octal numbers. Null or missing arguments are considered 0. The result of all expressions are strings, which represent decimal numbers. It is important to note that now two components of an expression can appear in the same word; except when next to components of expressions which are syntactically significant to the parser (`&` `!` `<` `>` `(` `)`), expression components should be surrounded by spaces.

Also available in expressions as primitive operands are command executions enclosed in `{` and `}` and file inquiries of the form `-l name` where *l* is one of:

```
r      Read access
w      Write access
x      Execute access
e      Existence
o      Ownership
z      Zero size
f      Plain file
d      Directory
```

The specified name is command and file name expanded and then tested to see if it has the specified relationship to the real user. If the file does not exist or is inaccessible, then all inquiries return false, that is, 0. Command runs succeed, returning true (1), if the command exits with status 0, otherwise they fail, returning false (0). If more detailed status information is required, run the command outside an expression and the examine `status` shell variable.

---

## Nonbuilt-in Command Execution

When a command to run is found not to be a built-in command, the shell attempts to run the command with *execve*. (See the **exec** system call in *AIX Operating System Technical Reference*.) Each word in the **path** shell variable names a directory from which the shell attempts to run the command. If it is given neither a **-c** nor a **-t** flag, the shell will hash the names in these directories into an internal table so it only tries an **exec** in a directory if there is a possibility that the command resides there. If this mechanism has been turned off with **unhash**, or if the shell is given a **-c** or **-t** (and in any case for each directory component of **path** that does not begin with a **/**), the shell concatenates with the given command name to form a path name of a file, which it then attempts to run.

Parenthesized commands always run in a subshell. Thus, `(cd ; pwd) ; pwd` displays the **home** directory without changing the current directory location, whereas `cd ; pwd` changes the current directory location to the **home** directory. Parenthesized commands are most often used to prevent **chdir** from affecting the current shell.

If the file has execute permissions, but is not an executable binary to the system, then it is assumed to be a file containing shell commands and a new shell runs to read it.

If there is an alias for shell, then the words of the alias will be prefixed to the argument list to form the shell command. The first word of the alias should be the full path name of the shell. Note that this is a special, late-occurring case of alias substitution and only allows words to be prefixed to the argument list without modification.

## Signal Handling

The shell normally ignores **QUIT** signals. Jobs running detached are immune to signals generated from the keyboard (**INTERRUPT**, **QUIT**, and **HANGUP**). Other signals have the values the shell inherited from its parent. You can control the shell's handling of **INTERRUPT** and **TERMINATE** signals in shell procedures with **onintr**. Login shells catch the **TERMINATE** signal; otherwise, this signal is passed on to children from the state in the shell's parent. In no case are **INTERRUPT**s allowed when a login shell is reading the **.logout** file.

## Limitations

The following are **csh** limitations:

- Words can be no longer than 1024 characters.
- Argument lists are limited to 5120 characters.
- The number of arguments to a command that involves file name expansion is limited to 1/6th the number of characters allowed in an argument list.
- Command substitutions can substitute no more characters than are allowed in an argument list.



- To detect looping, the shell restricts the number of alias substitutions on a single line to 20.

## Flags

If the first argument to the shell is - (minus), this is a login shell. The flags are interpreted as follows:

- c Reads commands from the (single) following argument, which must be present. Any remaining arguments are placed in **argv**.
- e Exits if any invoked command ends abnormally or yields a nonzero exit status.
- f Starts without searching for or running commands from the **.cshrc** file in the your home directory.
- i Prompts for its top-level input (an interactive shell), even if input does not appear to be coming from a work station. Shells are interactive without this flag if their input and output are attached to work stations.
- n Parses commands but does not run them. This aids in syntactic checking of shell procedures.
- s Takes command input from the standard input.
- t Reads and processes a single line of input. You can use a \ to escape the new-line character at the end of the current line to continue onto another line.
- v Sets the **verbose** shell variable, with the effect that command input is echoed after history substitution.
- V Sets the **verbose** shell variable even before **.cshrc** runs.
- x Sets the **echo** shell variable, so that commands are echoed immediately before they run.
- X Sets the **echo** shell variable even before **.cshrc** runs.

After processing of flag arguments, if arguments remain but none of the **-c**, **-i**, **-s**, or **-t** flags were given, the first parameter is taken as the name of a file of commands. The shell opens this file and saves its name for possible resubstitution by **\$0**. The shell runs a standard shell, if the first character of a procedure is not a **#**, that is, if the procedure does not start with a comment. Remaining parameters initialize the **argv** variable.

## Files

<code>\$HOME/.cshrc</code>	Read at beginning of execution by each shell.
<code>\$HOME/.login</code>	Read by login shell, after <b>.cshrc</b> at login.
<code>\$HOME/.logout</code>	Read by login shell, at logout.
<code>/bin/sh</code>	Standard shell.
<code>/tmp/sh*</code>	Temporary file for < <.

/etc/passwd      Source of home directories for  $\sim$ *name*.

## Related Information

The following commands: “**cd**” on page 121, “**make**” on page 474, “**pr**” on page 561, and “**sh**” on page 637.

The **access**, **exec**, **fork**, **pipe**, **umask**, and **wait** system calls, the **a.out** and **environ** files, and the **environment** miscellaneous facility in *AIX Operating System Technical Reference*.

“Overview of International Character Support” in *Managing the AIX Operating System*.

# csplit

---

## csplit

---

### Purpose

Splits files by context.

### Syntax



OL805177

### Description

The **csplit** command reads a *file* and separates it into segments defined by the specified parameters (*parm . . .*). By default, **csplit** writes these segments to files **xx00+ . . . xxn**, where *n* is the number of *parms* listed on the command line (*n* may not be greater than 99). These new files get the following pieces of *file*:

- 00:** From the start of *file* up to, but not including, the line referenced by the first *parm*.
- 01:** From the line referenced by the first *parm* up to the line referenced by the second *parm*.
- .
- .
- .
- n* + 1:** From the line referenced by the last *parm* to the end of *file*.

Note that **csplit** does not alter the original *file*.

The specified *parms* can be a combination of the following:

- /pattern/* Creates a file that contains the segment from the current line up to (but not including) the line containing *pattern*, which becomes the current line.
- %pattern%* Makes the line containing *pattern* the current line, but does not create a file for the segment.
- + num*  
*- num* Moves forward or backward the specified number of lines from the line matched by an immediately preceding *pattern* parameter (for example, */Page/-5*).

- linenum* Creates a file containing the segment from the current line up to (but not including) *linenum*, which becomes the current line.
- {*number*} Repeats the preceding argument the specified *number* of times. This number can follow any of the *pattern* or *linenum* parameters. If it follows a *pattern* parameter, **csplit** reuses that *pattern* the specified number of times. If it follows a *linenum* parameter, **csplit** splits the file from that point every *linenum* of lines for the specified number of times.

Quote all *pattern* parameters that contain blanks or other characters special to the shell. Patterns may not contain embedded new-line characters. In an expression such as [a-z], the minus means “through” according to the current collating sequence. A collating sequence may define *equivalence classes* for use in character ranges. See the “Overview of International Character Support” in *Managing the AIX Operating System* for more information on collating sequences and equivalence classes.

## Flags

- f *prefix* Specifies the *prefix* name for the created file segments. **xx** is the default *prefix*.
- k Leaves created file segments intact in the event of an error.
- s Suppresses the display of character counts.

## Examples

1. To split the text of a book into a separate file for each chapter:

```
csplit book "/^ *Chapter *[0-9]* *${10}"
```

This creates files named **xx00**, **xx01**, **xx02**, . . . ,**xx10**, which contain individual chapters of the file **book**. Each chapter begins with a line that contains only the word **Chapter** and the chapter number. The file **xx00** contains the front matter that comes before the first chapter. The {10} after the *pattern* allows up to 10 chapters.

2. To specify the prefix for the created file names:

```
csplit -f chap book "/^ *Chapter *[0-9]* *${10}"
```

This splits **book** into files named **chap00**, **chap01**, **chap02**, . . . ,**chap10**.

## Related Information

The following commands: “**ed**” on page 280, “**sh**” on page 637, and “**regcmp**” on page 595.

The **regxp** file in *AIX Operating System Technical Reference*.

“Overview of International Character Support” in *Managing the AIX Operating System*.

# ctab

---

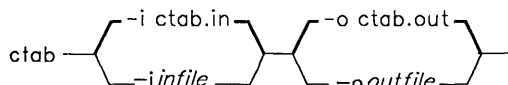
## ctab

---

### Purpose

Produces a collating table.

### Syntax



OL805451

### Description

The **ctab** command takes an input file (by default a file named **ctab.in** found in the current directory) and produces a binary file (by default named **ctab.out**) containing a collating table. These output files are stored in a conventional directory. Programs that need the current collating and case information use the **NLCTAB** environment variable to access that information.

The following conventions are used to make it easier to set up a table file:

- One line of information is present for each character explicitly named.
- A line beginning with the word **option** serves to change one or more of the default conditions or metacharacters built into **ctab**. An **option** line contains a set of name/value pairs, with each half of each pair delimited by tab or space characters. The following is a list of recognized names:

<b>eclass</b>	Turns the use of equivalence classes on or off globally. The assigned value must be <b>on</b> (the default) or <b>off</b> .
<b>sep</b>	Uses the assigned value as the field separator character. The default value is <b>:</b> (colon).
<b>trans</b>	Uses the assigned value of the “translate” indicator in subject character fields. The default character is <b> </b> (vertical bar).
<b>repeat</b>	Uses the assigned value as the “same as last line” indicator in subject character field. The default value is <b>^</b> (circumflex).
<b>comment</b>	Uses the assigned value as the comment character. The default value is the <b>#</b> character.

- The order of the per-character input lines specifies the collating sequence.

- By default, fields on a line are separated by colons. Tabs or spaces may surround fields or separators. You can change the separator character with an option line.
- Use an octal escape sequence in the ASCII range to name a nonprintable character. A backslash character that does not form part of a valid escape sequence serves to strip the following character, including a second backslash, of any special meaning it otherwise would have. For example, to include the colon character in the collating sequence, use the following line:

```
\::
```

The input file format includes a comment convention, namely that the remainder of the line following a # character is ignored. The comment character can be changed with an option line.

## Input File Specification

Use the following rules to build *infile*, entering field information for each line.:

1. The first field on a line contains the **subject character**, a character to be inserted into the collating sequence at that point.
  - This subject character definition can include a **translation mechanism**:
    - Instead of a single character, this field may contain two or more characters that are to be collated as a single unit, or
    - The single subject character may be followed by a vertical bar (!) and a single- or multiple-character string. The vertical bar indicates that the first character will be translated to the second string before being collated.  
 For example, to treat an “é” (e acute) as equivalent to the character “e,” use the following line:  
 ö!oe
    - One restriction is placed on the translation mechanism: the subject character cannot be contained in the translated string of characters. For example, the following line is illegal:  
 o!oe
  - Any form of the first field may contain a trailing circumflex (^) to indicate that the current character is to collate to the same value as the preceding one. However, a circumflex following a translation string is illegal because the subject character to be translated has no inherent collating value.
  - If the subject field contains a string of multiple characters (to collate as a unit), its first character must be declared elsewhere to establish the default collating sequence of that character.

- The translate and collating no-change characters can be changed with **option** lines.
2. The second and third fields specify whether or not a character is alphabetic and what its lower- and uppercase equivalents are:
    - If a subject character is to be treated as a lowercase alphabetic, the second field on its line is its uppercase equivalent, and the third field must be **l** or **L**.
    - If a subject character is to be treated as an uppercase alphabetic, the second field on its line is its lowercase equivalent, and the third field must be **u** or **U**.
    - If a subject character is to be treated as a control character or a space character, the third field must be **c**, **C**, **s**, or **S**.
    - Each character explicitly named whose line contains a nonnull second field will be considered alphabetic (that is, matched by **NCisalpha**). Characters that do not have an uppercase or lowercase equivalent (that is, that have a null second field) but that you wish to be considered alphabetic should simply contain a third field that is **l**, **L**, **u**, or **u**.
  3. The fourth field on a line is used explicitly to specify the first character in the **equivalence class** of the subject character. The members of one equivalence class must be consecutively listed in the input file.
    - There cannot be any gaps within a particular equivalence class. For example, the following lines will put the characters **a**, **b**, and **c** in the same equivalence class:  
a:A:l:a  
b:B:l:a  
c:C:l:a
    - As a convenience, if the fourth field is not specified, then the group of consecutive characters with blank fourth fields, provided that they are all based on the same Roman alphabetic character, will be placed in the same equivalence class. To reiterate, only characters with the same base will be placed into the same equivalence class by default. If you wish to have many characters from different bases belong to one equivalence class, as in the example above, the first character of the equivalence class has to be specified in the fourth field for every character specified.
    - It is illegal to specify an equivalence character that comes later in the collating sequence. The fourth field can refer only to characters that have already been mentioned.
    - All **international character support** characters not based on Roman alphabetic characters by default are the sole members of their equivalence class.

Characters not named in the table file that have an ordinal value (that is, a value as an **NLchar**) below the ordinal value of the lowest-valued character named are put into the collating sequence below the first character in the table file. All other characters not

named in the table file are put into the collating sequence above the last character in the table file.

The standard characters for decimal and hexadecimal digits are always marked as digits (to be matched by **NCisdigit** and **NCisxdigit**). All other printable characters not marked as alphabetic are marked as punctuation.

## Flags

- i** *infile* Specifies the name of the input file (**ctab.in** by default).
- o** *outfile* Specifies the name of the output file (**ctab.out** by default).

## Files

<code>/usr/lib/nls/ascii.in</code>	Input file listing the ASCII range of characters.
<code>/usr/lib/nls/iso.in</code>	Input file listing the ISO Collating Sequence

## Related Information

The **NCisalpha**, **NCisdigit**, **NCisxdigit**, **nls**, and **NLgetenv** subroutines in *AIX Operating System Technical Reference*.

The “Overview of International Character Support” in *IBM RT PC Managing the AIX Operating System*.



## ctags

---

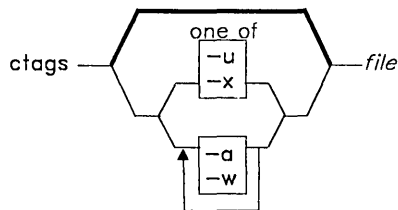
## ctags

---

### Purpose

Makes a tags file.

### Syntax



OL805457

### Description

The **ctags** command makes a tags file for **ex** and **vi** editors from the specified C, Pascal, and FORTRAN source files. A tags file gives the locations of specified objects (in this case functions) in a group of files. Each line of the tags file contains the object name, the file in which it is defined, and an address specification for the object definition. Functions are searched with a pattern. Specifiers are given in separate fields on the line, separated by blanks or tabs. Using the tags file, **ex** and **vi** can quickly find these object definitions.

If a file name ends in **.c** or **.h**, it is assumed to be a C source file and is searched for C routine and macro definitions. Others are first examined to see if they contain any Pascal or FORTRAN routine definitions; if not, they are processed again for C definitions.

The tag **main** is treated specially in C programs. The tag formed is created by prefixing **M** to the file name, removing a trailing **.c** (if any), and removing the leading path name components. This makes use of **ctags** practical in directories with more than one program.

**Note:** Recognitions of the keywords **function**, **subroutine**, and **procedure** in FORTRAN and Pascal code is performed in a very simple-minded way. No attempt is made to deal with block structure; if you have two Pascal procedures with the same name but in different blocks, **ctags** may yield inadequate results.

The **ctags** command does not know about **#ifdef**.

## Flags

- a Appends to tags file.
- w Suppresses warning diagnostics.
- x Causes **ctags** to display a list of object names, the line number and file name on which each is defined, as well as the text of that line. This provides a simple index index. If you specify this flag, **ctags** does not build a tags file.
- u Updates the specified files in tags; that is, all references to them are deleted, and the new values are appended to the file. This flag may be slow. (It is usually faster to simply rebuild the tags file.)

## Files

tags                      Output tags file

## Related Information

The following commands: “**ex**” on page 312 and “**vi, vedit, view**” on page 832.

# cut

---

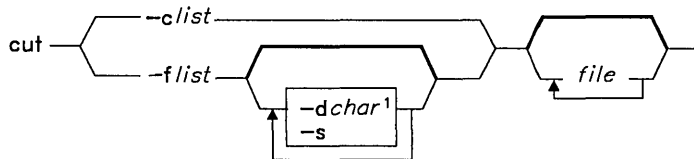
## cut

---

### Purpose

Writes out selected fields from each line of a file.

### Syntax



<sup>1</sup> The default *char* is a tab.

OL805178

### Description

The **cut** command cuts out columns from a table or fields from each line of a file and writes these columns or fields to standard output. If you do not specify a *file*, **cut** reads standard input.

You must specify either the **-c** or **-f** flag. The *list* parameter is a comma-separated and/or minus-separated list of integer field numbers (in increasing order). The minus separator indicates ranges. Some sample *lists* are 1,4,7; 1-3,8; -5,10 (short for 1-5,10); and 3- (short for third through last field). The fields specified by *list* can be a fixed number of character positions, or the length can vary from line to line and be marked with a field delimiter character, such as a tab character.

You can also use the **grep** command to make horizontal cuts through a file and the **paste** command to put the files back together. To change the order of columns in a file use **cut** and **paste**.

### Flags

**-clist** Specifies character positions. For example, if you specify **-c1-72**, **cut** writes out the first 72 characters in each line of the file. Note that there is no space between **-c** and *list*.

- 
- dchar** Uses the specified *character* as the field delimiter when you specify the **-f** flag. You must quote characters with special meaning to the shell, such as the space character.
  - flist** Specifies a list of fields assumed to be separated in the file by a delimiter character, by default the tab character. For example, if you specify **-f1,7**, **cut** writes out only the first and seventh fields of each line. If a line contains no field delimiters, **cut** passes them through intact (useful for table subheadings), unless you specify the **-s** flag.
  - s** Suppresses lines that do not contain delimiter characters (use only with the **-f** flag).

## Example

To display several fields of each line of a file:

```
cut -f1,5 -d: /etc/passwd
```

This displays the login name and full user name fields of the system password file. These are the first and fifth fields (**-f1,5**) separated by colons (**-d:**).

So, if the **/etc/passwd** file looks like this:

```
su:UHuj9Pgdvz0J":0:0:User with special privileges:/:/bin/sh
daemon*:1:1::/etc:
bin*:2:2::/bin:
sys*:3:3::/usr/src:
adm*:4:4:System Administrator:/usr/adm:/bin/sh
pierre:boodwqT3irHFE:200:200:Pierre Harper:/u/pierre:/bin/sh
joan:wijBNaYpCZuL.:202:200:Joan Brown:/u/joan:/bin/sh
```

then **cut** produces:

```
su:User with special privileges
daemon:
bin:
sys:
adm:System Administrator
pierre:Pierre Harper
joan:Joan Brown
```

## Related Information

The following commands: “**grep**” on page 381 and “**paste**” on page 547.

# cvid

---

## cvid

---

### Purpose

Creates a VRM install diskette for backup purposes.

### Syntax

```
cvid device [-f vrmmnt] [-v IBMVRM] [/vrm/vproto] [-f fs-ID] [-v vol-ID] [prototypefile]
```

OL805104

### Description

The **cvid** command backs up the VRM minidisk onto a diskette. Since you can reinstall the VRM system from this backup diskette, use **cvid** as a precautionary measure before modifying the VRM. You must be a member of the system group or operating with superuser authority to run this command.

The *device* parameter specifies the device (special file) to which **cvid** copies the VRM. This can be a block device name, a raw device name, or a directory name. If *device* is a directory name, **cvid** reads the */etc/filesystems* file for the corresponding device. **cvid** uses the *prototypefile* parameter to determine the size of the new file system. *prototypefile* defaults to */vrm/vproto*. For more information on prototype files, see “**mkfs**” on page 487 and “**proto**” on page 573.

### Flags

- f *fs-ID* Makes *fs-ID* the label for the new file system. The default label is **vrmmnt**.
- v *vol-ID* Makes *vol-ID* the volume label for the new file system. The default label is **IBMVRM**.

### Related Information

The following commands: “**mkfs**” on page 487 and “**mount**” on page 498.

---

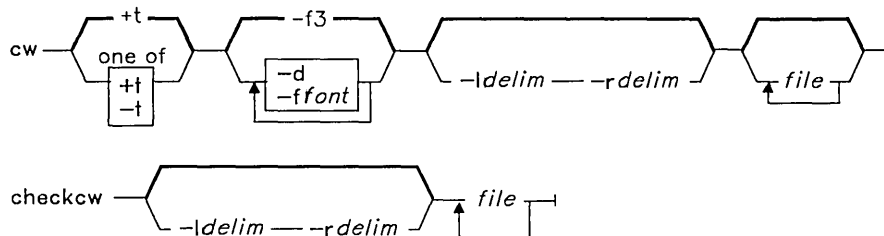
## cw, checkcw

---

### Purpose

Prepares constant-width text for **troff**.

### Syntax



OL805427

### Description

The **cw** command preprocesses **troff** files containing text to be typeset in the constant-width (CW) font. **cw** reads standard input if you do not specify a *file* or if you specify a `.` (minus) as one of the input file names. It writes its output to standard output.

Since the text that is typeset by **cw** resembles the output of line printers and work stations, it can be used to typeset examples of programs and computer output in user manuals and programming texts. It has been designed to be distinctive when used with the Times Roman font.

Because the CW font contains a “nonstandard” set of characters and because text typeset with it requires different character and interword spacing than is used for “standard fonts,” you must use **cw** to preprocess documents that use the CW font. The CW font contains the 94 printing ASCII characters:

```
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789
!$%&'()*+@.,/:;=?[]|_~`<>{}#\
```

plus eight non-ASCII characters represented by four-character **troff** strings (in some cases attaching these strings to “nonstandard” graphics):

Character	Symbol	Troff Name
"Cents" sign	¢	\(ct
EBCDIC "not" sign	¬	\(no
Left arrow	←	\(<-
Right arrow	→	\(>-)
Down arrow	↓	\(da
Vertical single quote	'	\(fm
Control-shift sign	␣	\(dg
Visible space sign	␣	\(sq
Hyphen	-	\(hy
Up arrow	↑	\(ua
Home arrow	↵	\(lh

OL805409

The **cw** command recognizes five request lines, as well as user-defined delimiters. The request lines look like **troff** macro requests. **cw** copies them in their entirety onto the output. Thus, you can define them as **troff** macros; in fact, the **.CW** and **.CN** macros *should* be so defined. The five requests are:

- .CW** Marks the start of text to be set in the CW font. This request causes a break. It can take the same flags (in the same format) as those available on the **cw** command line.
- .CN** Marks the end of text to be set in the CW font. This request causes a break. It can take the same flags (in the same format) as those available on the **cw** command line.
- .CD** Changes the delimiters and/or settings of other flags. It can take the same flags (in the same format) as those available on the **cw** command line. The purpose of this request is to allow the changing of flags other than at the beginning of a document.
- .CP *argument-list*** Concatenates all the arguments (delimited like **troff** macro arguments), with the odd-numbered arguments set in the CW font and the even-numbered ones in the prevailing font.
- .PC *argument-list*** Acts the same as **.CP**, except the even-numbered (rather than odd-numbered) arguments are set in CW font.

The **.CW** and **.CN** requests should bracket text that is to be typeset in the CW font “as is.” Normally, **cw** operates in the transparent mode. In that mode, every character between **.CW** and **.CN** request lines represents itself, except for the **.CD** request and the special four-character names listed previously. In particular, **cw** arranges for all periods (.) and apostrophes (') at the beginning of lines, and all backslashes (\) and ligatures (fi, ff, and so on) to be hidden from **troff**. The transparent mode can be turned off by using the **-t** flag, in which case normal **troff** rules apply. In either case, **cw** hides from the user the effect of the font changes generated by the **.CW** and **.CN** requests.

You can also use the `-l` and `-r` flags to define delimiters with the same function as the `.CW` and `.CN` requests. They are meant to enclose words or phrases that are to be set in CW font in the running text. `cw` treats text between delimiters as it does text bracketed by `.CW/.CN` pairs, with one exception. Spaces within `.CW/.CN` pairs have the same width as other CW characters, while spaces within delimited text are half as wide, so they have the same width as spaces in the prevailing text. Delimiters have no special meaning inside `.CW/.CN` pairs.

The `checkcw` command checks that left and right delimiters, and the `.CW/.CN` pairs are properly balanced. It prints out all lines in the section with the unmatched delimiters.

**Note:** It is unwise to use `.` (period) or `\` (backslash) as delimiter characters.

Certain CW characters do not combine well with certain Times Roman characters, for example, the spacing between a CW `&` (ampersand) followed by a Times Roman comma `(.)`. In such cases, using `troff` half-and quarter-space requests can help.

The `troff` code produced by `cw` is difficult to read.

The `mm` and `mv` macro packages contain definitions of `.CW` and `.CN` macros that are adequate for most use. If you define your own, make sure that the `.CW` macro invokes the `troff` no-fill (`.nf`) mode, and the `.CN` macro restores the fill mode (`.fi`), if appropriate.

When set in running text, the CW font is meant to be set in the same point size as the rest of the text. In displayed matter, on the other hand, it can often be profitably set one point smaller than the prevailing point size. The CW font is sized so that, when it is set in 9-point, there are 12 characters per inch.

Documents that contain CW text may also contain tables and equations. If this is the case, the order of preprocessing must be `cw`, `tbl`, and `eqn`. Usually, the tables will not contain any CW text, although it is possible to have elements in the table set in the CW font. Care must be taken that `cw` does not modify the `tbl` format information. Attempts to set equations in the CW font are not likely to be pleasing or successful.

In the CW font, overstriking is most easily accomplished with backspaces. Because spaces (and therefore backspaces) are half as wide between delimiters as inside `.CW/.CN` pairs, two backspaces are required for each overstrike between delimiters.



## Flags

- d** Displays the current flag settings on the standard error output in the form of **troff** comment lines. This flag is meant for debugging.
- ffont** Replaces *font* with the **cw** font (default = 3, replacing the bold font). **-f5** is commonly used for formatters that allow more than four simultaneous fonts. This flag is useful only on the command line.
- ldelim** Sets the left delimiter as the one-or two-character string *delim* The left delimiter is undefined by default.
- rdelim** Set the right delimiter as *delim* The right delimiter is undefined by default. The left and right delimiters may (but need not) be different.
- t** Turns the transparent mode *off*.
- +t** Turns the transparent mode *on* (this is the default).

## Files

`/usr/lib/font/ftCW` CW font-width table.

## Related Information

The following commands: “**eqn**, **neqn**, **checkeq**” on page 300, “**mmt**, **checkmm**” on page 495, “**tbl**” on page 739, and “**troff**” on page 526.

The **mm** and **mv** miscellaneous facilities in *AIX Operating System Technical Reference*.

---

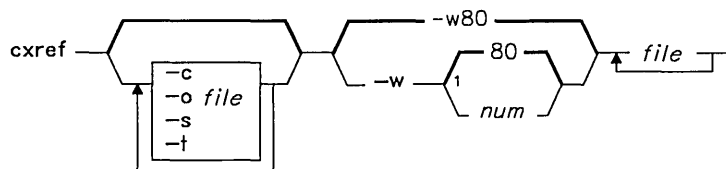
## cxref

---

### Purpose

Creates a C program cross-reference listing.

### Syntax



<sup>1</sup> Do not put a space between these items.

OL805180

### Description

The **cxref** command analyzes C program *files* and creates a cross-reference table, using a version of the **cpp** command to include **#define** directives in its symbol table. It writes to standard output a listing of all symbols in each file processed, either separately or in combination (see the **-c** flag on page 217). When a reference to a symbol is that symbol's declaration, an \* (asterisk) precedes it.

You can also use the **-D**, **-I**, and **-U** flags from the **cpp** command.

### Flags

- c** displays a combined listing of the cross-references in all input files.
- o file** Directs the output to the specified *file*.
- s** Does not display the input file names.
- t** Makes the listing 80 columns wide.
- w[num]** Makes the listing *num* columns wide, where *num* is a decimal integer greater than or equal to 51. If you do not specify *num* or if *num* is less than 51, the listing will be 80 columns wide.

## **cxref**

---

### **Files**

`/usr/lib/xcpp` Special version of C-preprocessor.

### **Related Information**

The following commands: “**cc**” on page 112 and “**cpp**” on page 163.

The discussion of **cxref** in *AIX Operating System Programming Tools and Interfaces*.

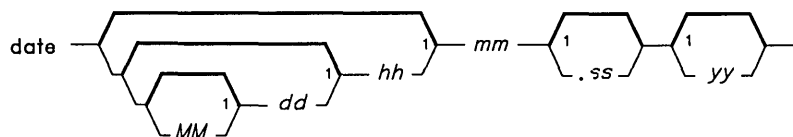
---

**date**

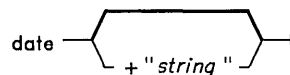

---

**Purpose**

Displays or sets the date.

**Syntax****Operating With Superuser Authority**

OL805105

**Operating Without Superuser Authority**

<sup>1</sup> Do not put a blank between these items.

OL805357

**Description**

**Warning:** Do not change the date while the system is running with more than one user.

If called with no flags or with a flag list that begins with a + (plus sign), the **date** command writes the current date and time to standard output. Otherwise, it sets the current date. Only a user operating with superuser authority can change the date and time. The **NLDATE** variable, if it is defined, controls the ordering of the day and month numbers in the date specifications. The default order is *MMddhhmm.ssyy* where:

- *MM* is the month number.
- *dd* is the number of the day in the month.
- *hh* is the hour in the day (using a 24-hour clock).
- *mm* is the minute number.
- *.ss* is the number of seconds.
- *yy* is the last two numbers of the year.

The alternative ordering is *ddMMhhmm.ssyy*.

## date

---

The current month, day, hour, and year are default values. The system operates in Greenwich Mean Time (GMT). **date** takes care of the conversion to and from local standard and daylight time as specified in the **NLTZ** environmental variable.

If you follow **date** with a + and a field descriptor, you can control the output of the command. You must precede each field descriptor with a percent sign (%). The system replaces the field descriptor with the specified value. Enter a literal % as %%. **date** copies any other characters to the output without change. **date** always ends the string with a new-line character. Output fields are fixed size (zero padded if necessary).

### Field Descriptors

- a** Displays the abbreviated day of the week (Sun to Sat or the non-English equivalent).
- d** Displays the day of month (01 to 31).
- D** Displays the date as *mm/dd/yy* (the default), or as *dd/mm/yy*. This format is specified by the **NLDATE** environment variable, if defined.
- h** Displays the abbreviated month (Jan to Dec or the non-English equivalent).
- H** Displays the hour (00 to 23).
- j** Displays the day of year (001 to 366).
- m** Displays the month of year (01 to 12).
- M** Displays the minute (00 to 59)
- n** Inserts a new-line character.
- r** Displays the time in AM/PM notation (or the non-English equivalent).
- S** Displays the second (00 to 59).
- t** Inserts a tab character.
- T** Displays the time as *hh:mm:ss* (the default), or as *mm:hh:ss*. This format is specified by the **NLTIME** environment variable, if defined.
- w** Displays the day of the week numerically (Sunday = 0).
- y** Displays the last two numbers of year (00 to 99).

### Examples

1. To display current date and time:  
date

2. To set the date and time:

```
date 02171425.45
```

This sets the date and time to 14:25:45 (45 seconds after 2:25 p.m.) February 17 of the current year.

3. To display the date and time in a specified format:

```
date +"%r %a %d %h %y (Julian Date: %j)"
```

This displays the date (assume current year is 1984) shown in Example 2 as:

```
02:25:03 PM Fri 17 Feb 84 (Julian Date: 048)
```

## Files

/dev/kmem

## Related Information

See the **time** and **stime** system calls and the **environment** miscellaneous facility in *AIX Operating System Technical Reference*.

“Overview of International Character Support” in *Managing the AIX Operating System*.

# dc

---

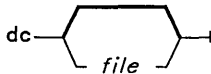
## dc

---

### Purpose

Provides an interactive desk calculator for doing arbitrary-precision integer arithmetic.

### Syntax



OL805106

### Description

The `dc` command is an arbitrary precision arithmetic calculator. `dc` takes its input from *file* or standard input until it reads an end-of-file character. It writes to standard output. It operates on decimal integers, but you may specify an input base, output base, and a number of fractional digits to be maintained. `dc` is structured overall as a stacking, reverse Polish, calculator.

The `bc` command (see page 95) is a preprocessor for `dc`. It provides infix notation and a syntax similar to the C language which implements functions and reasonable control structures for programs.

### Subcommands

<i>number</i>	Pushes the specified value onto the stack. A <i>number</i> is an unbroken string of the digits 0-9. To specify a negative number, precede it with <code>_</code> (underscore). A number may contain a decimal point.
<code>+ - / * % ^</code>	Adds (+), subtracts (-), multiplies (*), divides (/), remainders (%), or exponentiates (^) the top two values on the stack. <code>dc</code> pops the top two entries off the stack and pushes the result on the stack in their place. <code>dc</code> ignores fractional parts of an exponent.
<code>sx</code>	Pops the top of the stack and stores it in a register named <i>x</i> , where <i>x</i> may be any character.
<code>Sx</code>	Treats <i>x</i> as a stack. It pops the top of the main stack and pushes that value onto stack <i>x</i> .
<code>lx</code>	Pushes the value in register <i>x</i> on the stack. The register <i>x</i> is not changed. All registers start with zero value.

---

<b>Lx</b>	Treats <i>x</i> as a stack and pops its top value onto the main stack.
<b>d</b>	Duplicates the top value on the stack.
<b>p</b>	Displays the top value on the stack. The top value remains unchanged. The <b>p</b> interprets the top of the stack as an ASCII string, removes it, and displays it.
<b>P</b>	Interprets the top of the stack as a string, removes it, and displays it.
<b>f</b>	Displays all values on the stack.
<b>q</b>	Exits the program. If <b>dc</b> is executing a string, it pops the recursion level by two.
<b>Q</b>	Pops the top value on the stack and the string execution level by that value.
<b>x</b>	Treats the top element of the stack as a character string and executes it as a string of <b>dc</b> commands.
<b>X</b>	Replaces the number on the top of the stack with its scale factor.
<b>[string]</b>	Puts the bracketed <i>string</i> onto the top of the stack.
<b>&lt; x</b>	
<b>&gt; x</b>	
<b>= x</b>	Pops the top two elements of the stack and compares them. Evaluates register <i>x</i> as if it obeys the stated relation.
<b>v</b>	Replaces the top element on the stack by its square root. Any existing fractional part of the argument is taken into account, but otherwise the scale factor is ignored.
<b>!</b>	Interprets the rest of the line as a AIX command.
<b>c</b>	Cleans the stack: <b>dc</b> pops all values on the stack.
<b>i</b>	Pops the top value on the stack and uses that value as the number radix for further input.
<b>I</b>	Pushes the input base on the top of the stack.
<b>o</b>	Pops the top value on the stack and uses that value as the number radix for further output.
<b>O</b>	Pushes the output base on the top of the stack.
<b>k</b>	Pops the top of the stack, and uses that value as a nonnegative scale factor. The appropriate number of places displays on output and is maintained during multiplication, division, and exponentiation. The interaction of scale factor, input base, and output base is reasonable if all are changed together.
<b>z</b>	Pushes the number of elements in the stack onto the stack.



## dc

---

<b>Z</b>	Replaces the top number in the stack with the number of digits in that number.
<b>?</b>	Gets and runs a line of input.
<b>::</b>	<b>bc</b> uses these characters for array operations.

### Examples

1. To use **dc** as a calculator:

```
You: 1 4 / p
System: 0
You: 1 k      [ Keep 1 decimal place ]s.
      1 4 / p
System: 0.2
You: 3 k      [ Keep 3 decimal places ]s.
      1 4 / p
System: 0.250
You: 16 63 5 / + p
System: 28.600
You: 16 63 5 + / p
System: 0.235
```

You may type the comments (enclosed in [ ]s.), but they are provided only for your information.

When you enter **dc** expressions directly from the keyboard, press **Ctrl-D** to end the **bc** session and return to the shell command line.

2. To load and run a **dc** program file:

```
You: dc prog.dc
      5 lf x p  [ 5 factorial ]s.
System: 120
You: 10 lf x p [ 10 factorial ]s.
System: 3628800
```

This interprets the **dc** program saved in `prog.dc`, then reads from the work station keyboard. The `lf x` evaluates the function stored in register `f`, which could be defined in the program file `prog.c` as:

```
[ f: compute the factorial of n ]s.
[   (n = the top of the stack) ]s.
```

```
[ If 1>n do b; If 1<n do r ]s.
  [d 1 >b d 1 <r] sf
```

```
[ Return f(n) = 1          ]s.  
[d - 1 +] sb
```

```
[ Return f(n) = n * f(n-1) ]s.  
[d 1 - lf x *] sr
```

You can create **dc** program files with a text editor, or with the **-c** (compile) flag of the **bc** command.

When you enter **dc** expressions directly from the keyboard, press **Ctrl-D** to end the **bc** session and return to the shell command line.

## Related Information

The following command: “**bc**” on page 83.

“Overview of International Character Support” in *Managing the AIX Operating System*.

# dcopy

---

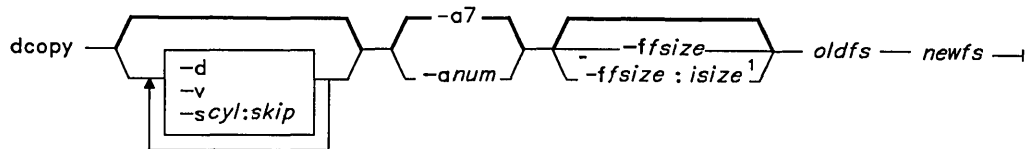
## dcopy

---

### Purpose

Copies file systems for the best access time.

### Syntax



<sup>1</sup> If not specified, the values from *oldfs* are used.

OL805420

### Description

**Warning:** *oldfs* and *newfs* must not refer to the same minidisk. Doing so will destroy the old file system.

The **dcopy** command copies an existing file system *oldfs* to a new file system *newfs*, appropriately sized to hold the reorganized results. For best results, *oldfs* should be the raw device and *newfs* should be the block device. If *oldfs* or *newfs* is a file system name, **dcopy** uses the corresponding block device given in `/etc/filesystems`. You should run **dcopy** on unmounted file systems (in the case of the root file system, copy to a new minidisk).

If you do not specify any flags, **dcopy** copies files from *oldfs*, compressing directories by removing vacant entries and spacing consecutive blocks in a file by the optimal rotational gap.

The **dcopy** command makes *newfs* identical to *oldfs* and preserves the pack and volume labels. Thus, to compress a file system without moving it, use the **dcopy** command to copy the file to another file system and the **dd** command to copy the file back.

The **dcopy** command catches **INTERRUPT** and **QUIT** signals and reports on its progress. To end **dcopy**, send a **QUIT** signal (**Ctrl-V**) and **dcopy** no longer catches **INTERRUPT** or **QUIT**. **dcopy** also attempts to modify its command line arguments so that its progress can be monitored with the **ps** command.

## Flags

- anum** Places files not accessed in the specified number of days after the free blocks of the destination file system. The default value of *num* is 7. If you do not specify *num*, no files are moved.
- d** Leaves the order of directory entries as is. If you do not specify this flag, **dcopy** moves subdirectories to the beginning of directories.
- ffsize[:isize]** Specifies the file system and i-node list sizes (in blocks). If not specified, the value from *oldfs* is used.
- scyl:skip** Supplies device information for creating the best organization of blocks in a file, where *cyl* is the number of block per cylinder and *skip* is the number of blocks to skip.
- v** Reports how many files were processed and how big the source and destination free lists are.

## Related Information

The following commands: “**fsck**, **dfscck**” on page 333, “**mkfs**” on page 487, and “**ps**” on page 579.

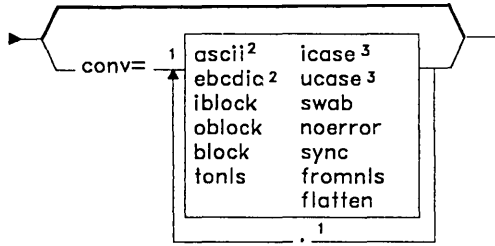
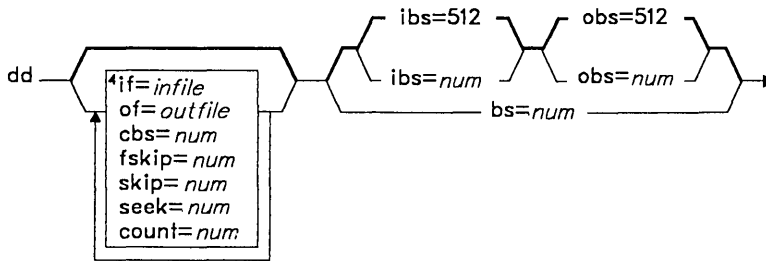
# dd

## dd

### Purpose

Converts and copies a file.

### Syntax



- <sup>1</sup> Do not put a blank between these items.
- <sup>2</sup> Use only one of `ascii` and `ebcdic`.
- <sup>3</sup> Use only one of `icase` and `ucase`.
- <sup>4</sup> `infile` and `outfile` default to standard input and standard output.

OL805373

### Description

The `dd` command reads the specified *infile* or standard input, does the specified conversions, and copies it to the specified *outfile* or standard output. The input and output block size may be specified to take advantage of raw physical I/O. The terms **block** and

*record* refer to the quantity of data read or written by **dd** in one operation and are not necessarily the same size as a disk block.

Where sizes are specified, a number of bytes is expected. A number may end with **w**, **b**, or **k** to specify multiplication by 2, 512, or 1024 respectively; a pair of numbers can be separated by an **x** to indicate a product.

The conversion requested by **conv=tonls** translates each extended character in a text file to a printable ASCII escape sequence that uniquely identifies the extended character. The complementary conversion, provided by **conv=fromnls**, translates ASCII escape sequences to the corresponding extended character. The conversion requested by **conv=flatten** translates an extended character to the single ASCII character most resembling it in appearance or to a ? (question mark) if no ASCII characters resemble that extended character.

The character set mappings associated with **conv=ascii** and **conv=ebcdic** are complementary operations, described in the **ebcdic** file in *AIX Operating System Technical Reference*. These attempt to map between ASCII and the subset of EBCDIC that is found on most terminals and keypunches.

The **cbs** specification is used only if the **ascii** or **ebcdic** conversion is specified. For ASCII conversions, **dd** places characters in a conversion buffer of size **cbs**, converts these characters to ASCII, trims trailing blanks and adds new-line characters before sending data specified output. For EBCDIC conversions, it places ASCII characters in the conversion buffer, converts these characters to EBCDIC, adds trailing blanks to create records of size **cbs**.

After it finishes, **dd** reports the number of whole and partial input and output blocks.

## Parameters

<b>if</b> = <i>infile</i>	Specifies the input file name; standard input is the default.
<b>of</b> = <i>outfile</i>	Specifies the output file name; standard output is the default.
<b>ibs</b> = <i>num</i>	Specifies the input block size in bytes; the default is 512.
<b>obs</b> = <i>num</i>	Specifies the output block size in bytes; the default is 512.
<b>bs</b> = <i>num</i>	Specifies both the input and output block size, superseding <b>ibs</b> and <b>obs</b> .
<b>cbs</b> = <i>num</i>	Specifies the conversion buffer size.
<b>skip</b> = <i>num</i>	Skip <i>num</i> input records before starting copy.
<b>seek</b> = <i>num</i>	Seek to the <i>num</i> th record from the beginning of output file before copying.
<b>fskip</b> = <i>num</i>	Skip past <i>num</i> end-of-file characters before starting copy; this parameter is useful for positioning on multi-file magnetic tapes.

## dd

---

<b>count = num</b>	Copies only <i>num</i> input blocks. The default block size is 512 bytes (see the <b>ibs</b> parameter).
<b>conv = spec[,spec . . . ]</b>	Specifies one or more of the following conversions:
<b>ascii</b>	Converts EBCDIC to ASCII.
<b>ebcdic</b>	Converts ASCII to EBCDIC.
<b>tonls</b>	Converts ASCII escape sequences to extended characters.
<b>fromnls</b>	Converts extended characters to ASCII escape sequences.
<b>flatten</b>	Converts extended characters to the ASCII character most resembling it or to a ? (question mark).
<b>iblock</b> <b>oblock</b> <b>block</b>	Minimizes data loss resulting from a read or write error on direct access devices. If you specify <b>iblock</b> and an error occurs during a block read (where the block size is 512 or the size specified by <b>ibs = num</b> ), <b>dd</b> attempts to reread the data block in smaller size units. If <b>dd</b> can determine the sector size of the input device, it reads the bad record one sector at a time. Otherwise, it reads it 512 bytes at a time. The input block size ( <b>ibs</b> ) must be a multiple of this “retry size.” This allows you to maximize disk input efficiency while ensuring that data loss associated with a read error is confined to a single sector. The <b>oblock</b> conversion works similarly on output. Specifying <b>block</b> is same as specifying <b>iblock,oblock</b> .
<b>lcase</b>	Makes all alphabetic characters lowercase.
<b>ucase</b>	Makes all alphabetic characters uppercase.
<b>swab</b>	Swaps every pair of bytes.
<b>noerror</b>	Does not stop processing on an error.
<b>sync</b>	Pads every input record to <b>ibs</b> .

**Note:** Normally, you need only write access to the output file. However, when the output file is not on a direct access device and you use the **seek** parameter, you also need read access to the file.

The **dd** command inserts new-line characters only when converting to ASCII; it pads only when converting to EBCDIC.

---

## Example

1. To convert an ASCII text file to EBCDIC:

```
dd if=text.ascii of=text.ebcdic conv=ebcdic
```

This converts `text.ascii` to EBCDIC representation, storing the EBCDIC version in `text.ebcdic`.

**Note:** When you specify `conv=ebcdic`, **dd** converts the ASCII `^` (circumflex) character to an unused EBCDIC character (9A hexadecimal), and ASCII `~` (tilde) to EBCDIC `¬` (NOT symbol).

2. To use **dd** as a filter:

```
li -l | dd conv=ucase
```

This displays a long listing of the current directory (`li -l`) in uppercase.

3. To read data that was written by a non-UNIX computer system:

```
dd if=/dev/rmt0 of=data ibs=800 cbs=80 conv=ascii,swab
```

This reads EBCDIC data from magnetic tape (`if=/dev/rmt0`) that has ten 80-byte card images per block. **dd** then converts it to ASCII (`conv=ascii`), swaps each pair of bytes (`conv=swab`), and stores it in a file named `data`.

Note that this example reads input from a raw device. The **dd** command can read and write blocks of any size, which makes it very useful for copying data to and from raw devices.

## Related Information

The following command: “**cp**” on page 156.

The `ebcdic` file in *AIX Operating System Technical Reference*.

The “Overview of International Character Support” in *Managing the AIX Operating System*.



# defkey

---

## defkey

---

### Purpose

Defines keyboard key assignments.

### Syntax

```
defkey [ -? ] [ file ]
```

OL805453

### Description

The **defkey** command lets you redefine the keyboard keys on the active virtual terminal. Input to **defkey** comes either interactively from the keyboard or from a redirected file. Key assignments can be a single character, non-spacing characters, or strings.

If you specify a *file* that does not exist, **defkey** creates and opens the file; if *file* exists, **defkey** opens the file. It then displays a menu that prompts you for input. This *file* can then be used as redirected input to **defkey**.

### Flags

-? Provides help information.

### Examples

1. To redefine a key or keys and create or add to a keyboard definition file:

```
defkey mykeys
```

This creates the file *mykeys* and prompts for input. When **defkey** ends, the keys that you specified will be redefined on the active virtual terminal. You can also use the file *mykeys* to redefine the keyboard on another virtual terminal with the command:

```
defkey < mykeys
```

2. To interactively redefine one or more keyboard keys for the active virtual terminal:

```
defkey
```

## **Related Information**

**hft** and **dispsym** in *AIX Operating System Technical Reference*.

*The Keyboard Description and Character Reference*.

# del

---

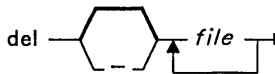
# del

---

## Purpose

Deletes files if the request is confirmed.

## Syntax



OL805049

## Description

The **del** command displays the list of specified *file* names and asks you to confirm your request to delete the group of files. To answer **YES** (delete the files), press the **Enter** key or enter a line beginning with **y**. Any other response specifies **NO** (do not delete the files).

The **del** command does not delete directories. See “**rmdir**” on page 605 for information about deleting directories.

**Warning:** The **del** command ignores file protection, allowing the owner of a file to delete a write-protected file. However, to delete a file, you must have write permission in the directory that the file exists in.

Since pressing the **Enter** key by itself is the same as answering “yes,” be careful not to delete files accidentally.

## Flag

- Requests confirmation for each specified *file* rather than for the entire group.

## Examples

1. To delete a file:  

```
del chap1.bak
```

This displays the message:

```
delete chap1.bak? (y)
```

to ask for confirmation before deleting chap1.bak. The (y) reminds you to press the **Enter** key or to enter y to answer “yes.”

2. To use **del** with pattern-matching characters:

```
del *.bak
```

Before passing the command line to **del**, the shell replaces the pattern \*.bak with the names of all the files in the current directory that end with .bak. (This is known as *file-name expansion*.) **del** asks for confirmation before deleting them all at one time.

3. To interactively select files to be deleted:

```
del - *
```

This displays the name of each file in the current directory one at a time, allowing you to select which ones to delete.

## Related Information

The following commands: “**rmdir**” on page 605 and “**rm**” on page 601.

# delta

---

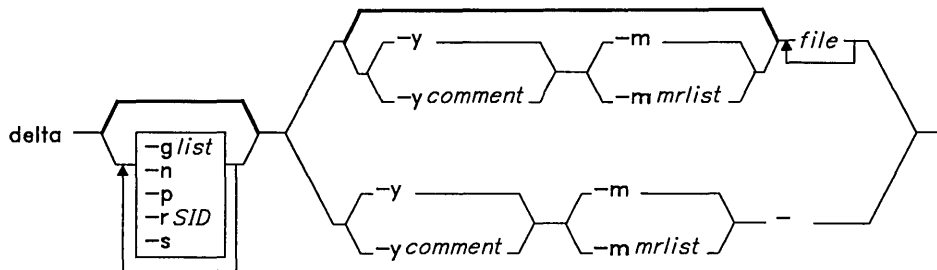
## delta

---

### Purpose

Creates a delta in a Source Code Control System file.

### Syntax



OL805056

### Description

The **delta** command is used to introduce into the named Source Code Control System (**SCCS**) *file* any changes that were made to the file version retrieved by a **get -e** command.

The **delta** command reads the *g-files* that correspond to the specified *files* (see “SCCS Files” on page 360) and creates a new delta.

If you specify a directory in place of *file*, **delta** performs the requested actions on all SCCS files within that directory (that is, on all files with the *s.* prefix). If you specify a - (minus) in place of *file*, **delta** reads standard input and interprets each line as the name of an SCCS file. When **delta** reads standard input, you must supply the **-y** flag. You must also supply the **-m** flag if the **v** header flag is set. (For more information on header flags, see the discussion in the **admin** command on page 54.) **delta** reads standard input until it reaches END OF FILE (**Ctrl-D**).

If you are not familiar with the delta numbering system, see *AIX Operating System Programming Tools and Interfaces* for more information.

**Note:** Lines beginning with an SOH ASCII character (binary 001) cannot be placed in the SCCS file unless the SOH is quoted using a \ (backslash). SOH has special meaning to SCCS and causes an error. See the *sccsfile* file in *AIX Operating System Technical Reference*.

---

A **get** of many SCCS files, followed by **delta** of those files, should be avoided when the **get** generates a large amount of data. Instead, you should alternate the use of **get** and **delta**.

## Flags

- glist** Specifies a list of *SIDs* (deltas) that are to be ignored when the **get** command creates the g-file. After you use this flag, **get** ignores this delta if it is one that it should not include when it builds the g-file.
- m[mrlist]** If the SCCS file has the **v** header flag set, then a Modification Request (MR) number must be supplied as the reason for creating the new delta.
- If you do not specify the **-m** flag, and the **v** header flag is set, **delta** reads MRs from the standard input. If standard input is a work station, **delta** prompts you for the MRs. **delta** continues to take input until it reads END OF FILE (**Ctrl-D**). It always reads MRs before the comments (see the **-y** flag). You can use blanks, tab characters, or both to separate MRs in a list.
- If the **v** header flag has a value, it is interpreted as the name of a program that validates the MR numbers. If **delta** returns a nonzero exit value from the MR validation program, **delta** assumes some of the MR numbers were invalid and stops running.
- n** Retains the g-file, which is normally removed at completion of **delta** processing.
- p** Writes to standard output (in the format of the **diff** command) the SCCS file differences before and after the delta is applied. See “**diff**” on page 246 for an explanation of the format.
- rSID** Specifies which delta is to be made to the SCCS file. You must use this flag only if two or more outstanding **get -e** commands were done on the same SCCS file by the same person. The *SID* can be either the *SID* specified on the **get** command line or the *SID* to be made, as reported by the **get** command (see Figure 2 on page 362 for additional information). An error results if the specified *SID* cannot be uniquely identified, or if a *SID* must be specified but it is not.
- s** Suppresses the information normally written to standard output on normal completion of the **delta** command.
- y[comment]** Specifies text used to describe the reason for making the delta. A null string is considered a valid *comment*. If your comment line includes special characters or blanks, the line must be enclosed in single or double quotation marks.
- If you do not specify **-y**, **delta** reads comments from standard input until it reads a blank line or END OF FILE (**Ctrl-D**). If input is from the keyboard, **delta** prompts for the comments. If the last character of a line is a backslash, it is ignored. Comments must be no longer than 512 characters.

## delta

---

### Example

To record changes you have made to an SCCS file:

```
delta s.prog.c
```

This adds a delta to the SCCS file `s.prog.c`, recording the changes made by editing `prog.c`. **delta** then asks you for a comment that summarizes the changes you made. Enter the comment, then press END OF FILE (Ctrl-D) or press the **Enter** key twice to indicate that you have finished the comment.

### Related Information

The following commands: “**admin**” on page 51, “**bdiff**” on page 88, “**cdc**” on page 123, “**get**” on page 359, “**help**” on page 391, “**prs**” on page 574, and “**rmdel**” on page 604.

The `scsfile` file in *AIX Operating System Technical Reference*.

The discussion of SCCS in *AIX Operating System Programming Tools and Interfaces*.

---

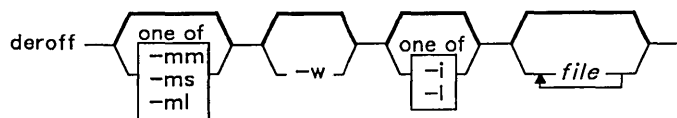
# deroff

---

## Purpose

Removes **nroff**, **troff**, **tbl**, and **eqn** constructs from files.

## Syntax



OL805181

## Description

The **deroff** command reads *files* (standard input by default), removes all **troff** requests, macro calls, backslash constructs, **eqn** constructs (between **.EQ** and **.EN** lines and between delimiters), and **tbl** descriptions (perhaps replacing them with blanks or blank lines), and writes the remainder of the file to standard output.

The **deroff** command normally follows chains of included files (**.so** and **.nx troff** commands). If a file has already been included, a **.so** naming it is ignored and a **.nx** naming that file ends execution.

**Note:** **deroff** is not a complete **troff** interpreter, so it can be confused by subtle constructs. Most errors result in too much rather than too little output.

The **-ml** flag does not handle nested lists correctly.

## Flags

- i** Suppresses the processing of included files.
- l** Suppresses the processing of included files whose names begin with **/usr/lib**, such as macro files in **/usr/lib/tmac**.
- mm** Ignores MM macros in text so that only running text is output (no text from macro lines is included).
- ml** Ignores MM macros in text (**-mm**) and also deletes MM list structures.
- ms** Ignores MS macros in text.



## deroff

---

- w** Makes the output a word list, with one word per line and all other characters deleted. In text, a word is any string that contains at least two letters and is composed of letters, digits, ampersands (&), and apostrophes ('). In a macro call, a word is a string that begins with at least two letters and contains a total of at least three letters. Delimiters are any characters other than letters, digits, apostrophes, and ampersands. Trailing apostrophes and ampersands are removed from words.

### Related Information

The following commands: “**eqn**, **neqn**, **checkeq**” on page 300, “**nroff**” on page 525, “**tbl**” on page 739, and “**troff**” on page 526.

---

## devices

---

### Purpose

Adds, deletes, changes, and displays device information.

### Syntax

devices —|

OL805306

### Description

The **devices** command lets you add, delete, change, or examine information about devices on the system. To use **devices** you must be a member of the system group or have superuser authority.

The **devices** command is an interactive, menu-driven program. For information on how to use it, see *Installing and Customizing the AIX Operating System*.

### Files

/dev	Directory.
/etc/filesystems	
/etc/predefined	
/etc/master	
/etc/system	
/etc/ports	
/etc/qconfig	
/tmp/CONFIGREPORT	
/etc/ddi	Directory.

### Related Information

The discussion of **devices** in *Installing and Customizing the AIX Operating System*.

**devnm**

---

**devnm**

---

**Purpose**

Names a device.

**Syntax**

```
devnm path
```

OL805114

**Description**

The **devnm** command reads *path*, identifies the special file associated with the mounted file system where *path* resides, and writes the special file name to standard output. Each *path* must be a full path name.

The most common use of the **devnm** command is by */etc/rc* to construct a mount table entry for the root device.

**Note:** This command is for local file systems only.

**Examples**

1. To identify the device on which a file resides:

```
devnm /diskette0/bob/textfile
```

This displays the name of the special device file on which */diskette0/bob/textfile* resides. If a diskette is mounted as **/diskette0**, then **devnm** displays:

```
fd0 /diskette0/bob/textfile
rfd0 /diskette0/bob/textfile
```

This means that */diskette0/bob/textfile* resides on the diskette drive **/dev/fd0**.

2. To identify the device on which a file system resides:

```
devnm /
```

This displays the name of the device on which the root file system (*/*) resides. The following list appears on the screen:

```
hd0 /
```

This means that `/` resides on `/dev/hd0`.

## Files

`/dev`            Directory.  
`/etc/mnttab`

## Related Information

The following commands: “`rc`” on page 594 and “`setmnt`” on page 635.

# df

---

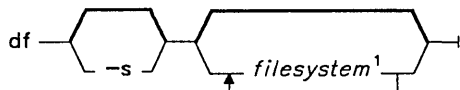
## df

---

### Purpose

Reports number of available disk blocks.

### Syntax



<sup>1</sup>The default action is to provide information for each file system in `/etc/filesystems` with the attribute `free=true`.

OL805052

### Description

The **df** command writes to standard output information about total space and available space on the specified file systems. *filesystem* can be the name of the device on which the file system resides or the directory on which it is mounted. If you do not specify *filesystem*, **df** uses each file system in `/etc/filesystems` that has the attribute `free=true`.

Normally, **df** uses free counts maintained in the superblock. Under certain exceptional circumstances, these counts may be in error.

If a file system is being actively modified at the instant **df** is run, the free count may be inaccurate.

### Flag

- s Checks for count errors by forcing **df** to fully search the free lists to verify the counts. The **df** command requires considerably more processing time when the `-s` flag is specified.

---

## Examples

1. To list information about all default file systems:

```
df
```

If your system is configured so that the `/`, `/usr`, `/u`, and `/tmp` directories reside in separate file systems, the output from the `df` command looks something like this:

Device	Mounted on	total	free	used	ifree	used
/dev/hd0	/	19368	9976	48%	4714	5%
/dev/hd1	/usr	24212	4808	80%	5031	19%
/dev/hd2	/u	9744	9352	4%	1900	4%
/dev/hd5	/tmp	3868	3856	0%	986	0%

2. To list information about the file system on a diskette:

```
df /dev/fd0
```

3. To list information about the file system normally mounted as `/diskette0`:

```
df /diskette0
```

## Files

```
/etc/filesystems
```

## Related Information

The following command: “**fsck**, **dfsck**” on page 333.

The `filesystem` file in *AIX Operating System Technical Reference*.

The discussion of `df` in *Managing the AIX Operating System*.

# diff

---

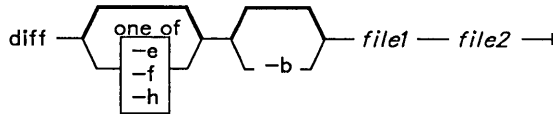
## diff

---

### Purpose

Compares text files.

### Syntax



OL805046

### Description

The **diff** command compares *file1* and *file2* and writes to standard output information about what changes must be made to bring them into agreement. If you specify a - (minus) for *file1* or *file2*, **diff** reads standard input. If *file1* is a directory, then **diff** uses a file in that directory with the name *file1*. If *file2* is a directory, then **diff** uses a file in that directory with the name *file1*.

The normal output contains lines of these forms:

Lines Affected in <i>file1</i>	Action	Lines Affected in <i>file2</i>
<i>num1</i>	<b>a</b>	<i>num2</i> [, <i>num3</i> ]
<i>num1</i> [, <i>num2</i> ]	<b>d</b>	<i>num3</i>
<i>num1</i> [, <i>num2</i> ]	<b>c</b>	<i>num3</i> [, <i>num4</i> ]

These lines resemble **ed** subcommands to convert *file1* into *file2*. The numbers before the action letters pertain to *file1*; those after pertain to *file2*. Thus, by exchanging **a** for **d** and reading backward, you can also tell how to convert *file2* into *file1*. As in **ed**, identical pairs (where *num1* = *num2*) are abbreviated as a single number.

Following each of these lines, **diff** displays all lines affected in the first file preceded by a `<`, then all lines affected in the second file preceded by a `>`.

Except in rare circumstances, **diff** finds a smallest sufficient set of file differences. An exit value of 0 indicates no differences, 1 indicates differences found, and 2 indicates an error.

**Note:** Editing scripts produced by the `-e` or `-f` flags cannot create lines consisting of a single period (`.`).

---

## Flags

- b Ignores trailing spaces and tab characters and considers other strings of blanks to compare as equal.
- e Produces output in a form suitable for use with the **ed** command to convert *file1* to *file2*.
- f Produces output in a form not suitable for use with **ed**, showing the modifications necessary to convert *file1* to *file2* in the reverse order of that produced under the **-e** flag.
- h Performs a faster comparison. This flag only works when the changed sections are short and well separated, but it does work on files of any length. The **-e** and **-f** flags are not available when you use the **-h** flag.

## Examples

1. To compare two files:

```
diff chap1.bak chap1
```

This displays the differences between the files `chap1.bak` and `chap1`.

2. To compare two files, ignoring differences in the amount of white space:

```
diff -b prog.c.bak prog.c
```

If two lines differ only in the number of blanks and tabs between words, then **diff** considers them to be the same.

3. To create a file containing commands that **ed** can use to reconstruct one file from another:

```
diff -e chap2 chap2.old >new.to.old.ed
```

This creates a file named `new.to.old.ed` that contains the **ed** commands to change `chap2` back into the version of the text found in `chap2.old`. In most cases, `new.to.old.ed` is a much smaller file than `chap2.old`. You can save disk space by deleting `chap2.old`, and you can reconstruct it at any time by entering:

```
(cat new.to.old.ed ; echo '1,$p') | ed - chap2 >chap2.old
```

The commands in parentheses add `1,$p` to the end of the editing commands sent to **ed**. The `1,$p` causes **ed** to write the file to standard output after editing it. This modified command sequence is then piped to **ed** (`| ed`), and the editor reads it as standard input. The `-` flag causes **ed** not to display the file size and other extra information since it would be mixed with the text of `chap2.old`. See page 654 for details about grouping commands with parentheses.



## diff

---

### Files

/tmp/d???? Temporary files.  
/usr/lib/diffh For the **-h** flag.

### Related Information

The following commands: “**bdiff**” on page 88 “**cmp**” on page 138, “**comm**” on page 144, “**ed**” on page 280, and “**sdiff**” on page 627.

---

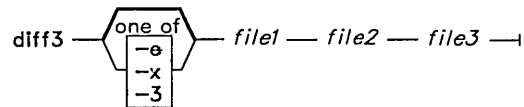
# diff3

---

## Purpose

Compares three files.

## Syntax



OL805053

## Description

The **diff3** command reads three versions of a file and writes to standard output the ranges of text that differ, flagged with the following codes:

```
==== All three files differ.
====1 file1 differs.
====2 file2 differs.
====3 file3 differs.
```

The type of change needed to convert a given range of a given file to match another file is indicated in one of these two ways in the output:

```
file : n1 a      Text is to be added after line number n1 in file, where file is 1, 2, or 3.
file : n1[,n2] c  Text in the range line n1 to line n2 is to be changed. If n1 = n2, the range may be abbreviated to n1.
```

The original contents of the range follows immediately after a **c** indication. When the contents of two files are identical, **diff3** does not show the contents of the lower-numbered file, although it shows the location of the identical lines for each.

**Note:** Editing scripts produced by the **-e** flag cannot create lines consisting only of a single period (.).

The **diff3** command does not work on files longer than 64K bytes.

## diff3

---

### Flags

- e Creates an edit script for use with the **ed** command to incorporate into *file1* all changes between *file2* and *file3* (that is, the changes that normally would be flagged `====` and `===3`).
- x Produces an edit script to incorporate only changes flagged `====`.
- 3 Produces an edit script to incorporate only changes flagged `===3`.

### Example

To list the differences among three files:

```
diff3 fruit.a fruit.b fruit.c
```

If *fruit.a*, *fruit.b*, and *fruit.c* contain the following data:

fruit.a	fruit.b	fruit.c
banana	apple	grape
grape	banana	grapefruit
kiwi	grapefruit	kiwi
lemon	kiwi	lemon
mango	orange	mango
orange	peach	orange
peach	pear	peach
pare		pear

then the output from **diff3** shows the differences between these files as follows. (The comments on the right do not appear in the output.)

```
==== • All three files are different.
1:1,2c - Lines 1 and 2 of the first file, fruit.a
  banana
  grape
2:1,3c - Lines 1 through 3 of fruit.b
  apple
  banana
  grapefruit
3:1,2c - Lines 1 and 2 of fruit.c
  grape
  grapefruit
===2 • The second file, fruit.b, is different.
```

```
1:4,5c  - Lines 4 and 5 are the same in fruit.a and fruit.c.
2:4a    - To make fruit.b look the same, add text after line 4.
3:4,5c
  lemon
  mango
====1   • The first file, fruit.a, is different.
1:8c
  pare
2:7c    - Line 7 of fruit.b and line 8 of fruit.c are the same
3:8c
  pear
```

## Files

```
/tmp/d3*
/usr/lib/diff3prog
```

## Related Information

The following command: “**diff**” on page 246.

# diffmk

---

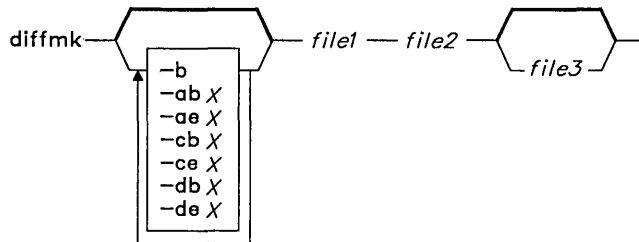
## diffmk

---

### Purpose

Marks differences between files.

### Syntax



OL805057

### Description

The **diffmk** command compares *file1* and *file2* and creates a third file that includes **change mark commands** for the **nroff** and **troff** commands. *file1* and *file2* are the old and new versions of the file. **diffmk** writes the newly created file to *file3*, if specified, or to standard output. This file contains the lines of *file2* with formatter change mark (**.mc**) requests inserted as appropriate. When *file3* is formatted, the changed or inserted text is marked by a | (vertical bar) at the right margin of each line. An \* (asterisk) in the margin indicates that a line was deleted.

If the environment parameter **DIFFMARK** is defined, it names a command string that **diffmk** uses to compare the files. (Normally, **diffmk** uses the **diff** command.) For example, you might set **DIFFMARK** to **diff -h** in order to better handle extremely large files.

### Flags

- abX** Uses *X* to mark where added lines begin.
- aeX** Uses *X* to mark where added lines end.
- b** Ignores differences that are only changes in tabs or spaces on a line.
- cbX** Uses *X* to mark where changed lines begin.

- ceX** Uses *X* to mark where changed lines end.
- dbX** Uses *X* to mark where deleted lines begin.
- deX** Uses *X* to mark where deleted lines end.

## Examples

1. To mark the differences between two versions of a text file:

```
diffmk chap1.old chap1 > chap1.nroff
```

This produces a copy of `chap1` containing **nroff/troff** change mark commands to identify text that has been added to, changed in, or deleted from `chap1.old`. This copy is saved in the file `chap1.nroff`.

2. To mark differences with non-**nroff/troff** messages:

```
diffmk -ab'>>New:' -ae'<<End New' chap1.old chap1 >chap1.nroff
```

This causes **diffmk** to write `>>New:` on the line before a section of new lines that have been added to `chap1` and to write `<<End New` on the line following the added lines. Changes and deletions still generate **nroff/troff** commands to put a `|` or `*` in the margin.

3. To use different **nroff/troff** marking commands and ignore changes in white space:

```
diffmk -b -cb'.mc %' chap1.old chap1 > chap1.nroff
```

This imbeds commands that mark changes with `%`, additions with `|`, and deletions with `*`. It does not mark changes that only involve a different number of spaces or tabs between words (`-b`).

## Related Information

The following commands: “**diff**” on page 246, “**nroff**” on page 525, and “**troff**” on page 526.

# dircmp

---

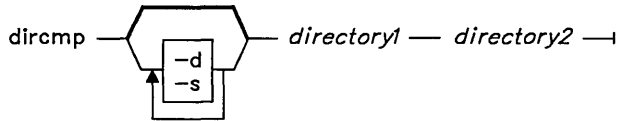
## dircmp

---

### Purpose

Compares two directories and the contents of their common files.

### Syntax



OL805004

### Description

The **dircmp** command reads *directory1* and *directory2* and writes information about their contents to standard output. First, **dircmp** compares the file names in each directory. When the same file name appears in both, **dircmp** compares the contents of both files.

In the output, **dircmp** lists the files unique to each directory. It then lists the files with identical names in both directories, but with different contents. With no flag, it also lists files that have identical contents as well as identical names in both directories.

### Flags

- d Displays for each common file name both versions of the differing file lines. The display format is the same as that of “**diff**” on page 246.
- s Does not list the names of identical files.

### Examples

1. To summarize the differences between the files in two directories:

```
dircmp proj.ver1 proj.ver2
```

This displays a summary of the differences between the directories *proj.ver1* and *proj.ver2*. The summary lists separately the files found only in one directory or the other, and those found in both. If a file is found in both directories, **dircmp** notes whether or not the two copies are identical.

2. To show the details of the differences between files:

```
dircmp -d -s proj.ver1 proj.ver2
```

The **-s** flag suppresses information about identical files. The **-d** flag displays a **diff** listing for each of the differing files found in both directories.

## Related Information

The following commands: “**cmp**” on page 138 and “**diff**” on page 246.





-u *file*      Writes records to *file* of files that are charged to no one. Records consist of the special file name, the i-node number, and the user ID.

-v              Writes a list to standard error of all files that are charged to no one.

The output of **diskusg** is normally the input to **acctdisk**, which generates total accounting records that can be merged with other accounting records. **diskusg** is normally run in **dodisk** (see “**acct/\***” on page 31).

## Examples

The following will generate daily disk accounting information:

```
for i in /dev/hd0 /dev/hd1 /dev/hd2 /dev/hd3
do
    diskusg $i > dtmp.'basename $i' &
done
wait
diskusg -s dtmp.* | sort +0n +1 | acctdisk > dacct
```

## Files

/etc/passwd      Used for user ID to login name conversions.

## Related Information

The following commands: “**acct/\***” on page 31, “**acctcms**” on page 36, “**acctcom**” on page 38, “**acctcon**” on page 42, “**acctmerg**” on page 46, “**acctprc**” on page 48, “**fwtmp**” on page 345, and “**runacct**” on page 606.

The **acct** system call and the **acct** and **utmp** files in *AIX Operating System Technical Reference*.

The discussion of accounting in *Managing the AIX Operating System*.

# display

---

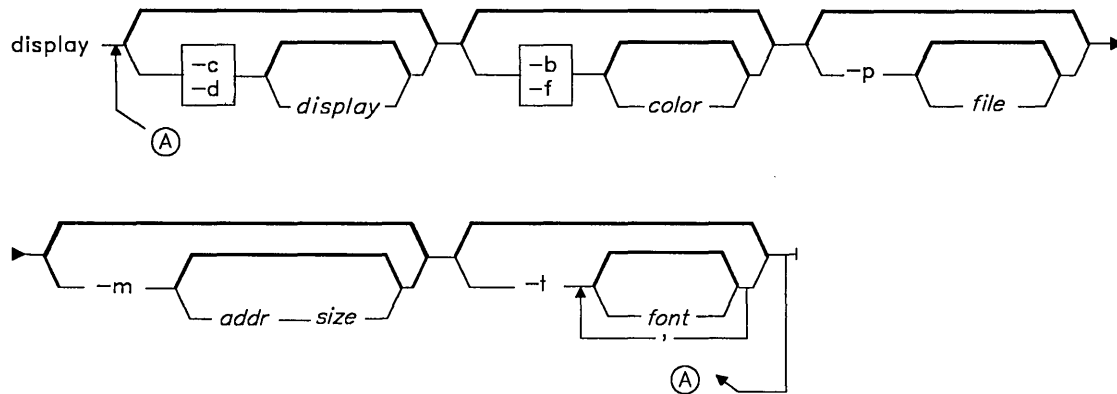
## display

---

### Purpose

Selects the physical display that an existing or new virtual terminal uses and sets colors and fonts.

### Syntax



OL805442

### Description

The **display** command changes the physical display assigned to the current virtual terminal or assigns a default display to be used when you open a virtual terminal. It also sets the foreground and background colors, the active color palette, and the active and alternate fonts on the current display. The *display* parameter can be one of the following names:

<b>pcmono</b>	PC Monochrome Adapter and Display
<b>egamono</b>	Enhanced Graphics Adapter and PC Monochrome Display
<b>egacol</b>	Enhanced Graphics Adapter and Display
<b>advmono</b>	Advanced Monochrome Graphics Adapter and Display
<b>advcol</b>	Advanced Color Graphics Adapter and Display
<b>extmono</b>	Extended Monochrome Graphics Adapter and Display
<b>megapel</b>	Megapel Display Adapter and IBM 5081 Display Models 16 and 19.

You can request only those displays that are actually installed on the system. If you have more than four, only four will be displayed on the **-c** and **-d** menus. Before **display** makes any changes, it checks all arguments for errors and, if it encounters one, displays a list of valid arguments and exits.

**Note:** You must insure that the **TERM** shell variable contains the proper value for whatever the current display is. See “**termdef**” on page 748 and the **terminfo** file in *AIX Operating System Technical Reference* for a list of these values.

## Flags

- b** [*color*]           Selects the background color. The *color* parameter is an integer from 1 to 8 for the Enhanced Graphics Display and from 1 to 16 for other color graphics displays. These values correspond to the first eight or sixteen entries in the active color palette (see the **-p** flag). For example, **-b 5** selects the fifth entry. If you do not specify a number, **display** lists the palette of active background colors and prompts you to select a number for the new background color.
- c** [*display*]           Changes the display used by the current virtual terminal.
- If you do not specify a *display*, you are given a menu of available options. This menu consists of a numbered list of display names and descriptions. The display number reflects the number of physical displays installed and their relative positions in the Real Screen Table. The current default is always display number 1 in this list. Changing the default alters the display number associated with each physical display. If the virtual terminal does not know the display/adaptor combination, the Name column will contain the words Unknown Display or ??????. A prompt at the bottom of the display list asks you to enter the new display number for the current or default display setting. Whenever you change the current display, the screen of that display clears.
- d** [*display*]           Changes the default display used when a virtual terminal is opened. If you do not specify a *display*, you are given a menu of available options (see the **-c** flag).
- f** [*color*]           Selects the foreground color. The *color* parameter is an integer from 1 to 16. These values correspond to the first sixteen entries in the active color palette (see the **-p** flag). If you do not specify a color number, **display** lists the palette of active foreground colors and prompts you to select a number for the new foreground color.
- m** [*addr size*]       Changes the DMA pinned page at the specified starting address to *size* 256K blocks. If you do not specify an address and a size, the current starting address and size is displayed.

## display

---

- p** [*file*]  
Changes the active color palette. The optional *file* parameter is the full path name to a file that contains a list of colors for the current display, one color per line, where each color is the decimal representation of the 32 bit color value. The color palette file can also contain blank lines and comment lines (a comment line must begin with a \* character in column one). Each supported display has a corresponding color file which contains its default active color palette. The name of this file is */etc/vtm/pal.name* where *name* is the display name described on page 258. This is the default value for the *file* parameter.
- t** [*font[,font] . . .*]  
Selects the primary and active alternate fonts for the current virtual terminal on the current display. The first *font* named in the optional list following **-t** will be the primary font. The remaining fonts will be alternates, in the order listed, for the active font table. If you do not specify eight font IDs, the first font will be used to fill out in the remaining entries in the active font table.

**Note:** All of the fonts in the list must be of the same size.

Some applications that use the **terminfo** file expect the italic font to be the first alternate and the bold font to be the second alternate fonts (see the **terminfo** file in *AIX Operating System Technical Reference* for more information).

If you do not specify any fonts, all of the fonts available for the current display will be listed, and you will be prompted first for the desired primary font ID and then for alternate font IDs until you enter F. As you enter alternate fonts, the **display** command checks that they are the same size as the new primary font. If you enter fewer than eight fonts, the primary font will be repeated in the remaining entries of the active font table.

You can specify combinations of the same flags on a single command line. **display** processes **-c** and **-d** flags first. If you specify **-c**, you will see the message Changing to current display . . . , and the current display will be changed. Any menu interface for the color or font parameters will be displayed there. A **-p** flag will be processed next. The screen will be immediately redrawn with the colors from the new color palette. Then any foreground, background, or font flags will be processed.

## Examples

1. To change the current virtual terminal display:

```
display -c egamono
```

This changes the display to the Enhanced Graphics Adapter and PC Monochrome Display.

2. To make the Advanced Color Graphics Display the default virtual terminal display:

```
display -d advcol
```

3. To change both the current and the default displays:

```
display -c pcmmono -d egacol
```

This makes the PC Monochrome Adapter and Display the current display and makes the Enhanced Graphics Adapter and Display the default display.

4. To change the active color palette for the current display:

```
display -p /u/new/palette
```

## Related Information

The following commands: “**open**” on page 541 and “**termdef**” on page 748.

The **terminfo** file in *AIX Operating System Technical Reference*.

“Using Display Station Features” in *IBM RT PC Using the AIX Operating System* and  
“Managing Display Station Features” in *IBM RT PC Managing the AIX Operating System*.

The default color palettes in *Virtual Resource Manager Technical Reference*.



- The working directory
- Each directory in the **dos** path.

When you enter a command, **dos** searches each directory for a file with a name composed of the command name and either the extension **.BAT**, the extension **.bat**, or no extension. If the file has the extension **.BAT** or **.bat**, it runs as a batch file. Otherwise, it runs as an AIX program. If it is a AIX program, it can be either a compiled program or a shell file. In either case you must have execute access to it.

The **dos** command supports two types of file systems: AIX file systems and DOS file systems. Each **dos** minidisk can contain either an AIX-formatted file system or a DOS-formatted file system. However, diskette drives (such as **/dev/fd0**) may contain only DOS-formatted file systems, unless the device is mounted as an AIX file system before you invoke **dos**.

**Warning:** Only one user or process at a time can access a **dos** file system. If a **dos** file system resides on a minidisk, two or more users may attempt to access the minidisk at the same time. Because **dos** has no way to warn you that another process is using a minidisk, you should allocate minidisks containing **dos** file systems on a per-user basis.

If a coprocessor on the system accesses a **dos**-formatted minidisk at the same time as an RT PC process, there is no conflict because only the first process has read/write privileges. Subsequent opens at the device level are limited to read-only access.

There are different restrictions for file names on DOS drives and AIX drives. For DOS Services drives:

- File names cannot be longer than 12 characters.
- The name is always stored in uppercase.
- All files in the directory must have unique names.
- There can be only one period in a file name.

For AIX file systems:

- File names cannot be longer than 14 characters.
- Names may contain either uppercase or lowercase letters.
- Two files in the same directory can have the same name if the letter case is different.
- There can be more than one period in a file name.
- All files in the directory must have unique names.

On AIX drives, file names that begin with a period specify hidden files. On DOS Services drives, hidden files have a bit set in the attribute byte of the file directory.

There are differences between AIX and DOS Services file formats. AIX ASCII files and DOS Services ASCII files are similar and can be converted from one format to the other. Two new commands, **FILETYPE** and **CONVERT**, are available for detecting and changing a file format.



## DOS Services Commands and Programs

There are several differences between the set of supported DOS Services commands and DOS commands.

### Unsupported DOS Commands and Programs

You can use all of the standard DOS commands except **BREAK**, **CTTY**, **EDLIN**, **EXE2BIN**, **GRAPHICS**, and **SYS**.

### Modified DOS Commands

The following DOS Services commands behave differently than the corresponding standard DOS commands:

- backup**     The **/M** parameter is not valid for DOS Services file systems.
- chdir**     Unlike DOS, DOS Services may not allow you to change to the highest directory in the file system.
- date**     This command lets only superuser change the date.
- dir**     Does not list file-name extensions in a separate column when executed on an AIX drive.
- format**     The **/B** is not supported. Two additional flags, **/U** and **/H** are supported. Use the **/U** flag to format a AIX diskette. Use the **/H** flag to format a fixed disk to contain DOS Services file systems in a single partition.
- Note:** The **format** command makes use of the **mksf** command, which in turn uses the **/etc/filesystems** file. If you modify this file, it will affect the **format** command.
- label**     On an AIX-formatted drive, the label is written to a file called **LABEL.VOL**. Reading a label is accomplished by reading this file. Changing a label modifies the contents of this file.
- Note:** The command **del \*.\*** deletes the volume label.
- mode**     Only option 3 (for an asynchronous communications adapter) is supported.
- print**     The DOS Services version does not ask you which device to store the print queue on. This information is set up in your user profile.
- The **/B**, **D**, **M**, **/S**, **/Q**, and **/U** configuration flags are not supported.
- set**     A **/U** flag lets you display the AIX environment as it is inherited by the **dos** command. You can change the environment variables internal to **dos**. When you exit from **dos**, the environment variables remain unchanged.
- time**     Allows only the superuser to change the time.

---

## Additional Commands

In addition to DOS commands, the following commands are available:

<b>!</b>	(The Escape command.) Runs the remainder of the command line as an AIX shell command.
<b>COMMAND</b>	The new flags which have been added to <b>dos</b> also apply to this command.
<b>CONVERT</b>	Converts a DOS format ASCII file to a AIX format ASCII file or a AIX format ASCII file to a DOS format ASCII file.
<b>ed</b>	Starts the line editor.
<b>EXIT</b>	Ends DOS Services. You can also use END OF FILE ( <b>Ctrl-D</b> ).
<b>FILETYPE</b>	Attempts to determine the format (AIX or DOS) and contents of the specified file.
<b>shutdown</b>	Provides for an orderly exit from the system.

## Flags

<b>-a</b>	Does not run the <b>AUTOEXEC.BAT</b> file.
<b>-c cmd</b>	Runs the specified command.
<b>-n</b>	Reads commands but does not run them.
<b>-v</b>	Displays the commands and their flags as they are read.
<b>-x</b>	Displays the commands and their flags as they are run.

## Files

<code>/usr/dos/bin/*</code>	DOS Services external commands.
<code>AUTOEXEC.BAT</code>	
<code>autoexec.bat</code>	

## Related Information

The following commands: “**dosdel**” on page 266, “**dosread**” on page 269 and “**doswrite**” on page 271.

The **dosinit** subroutine in *AIX Operating System Technical Reference*.

The discussion of **dos** in *Using AIX Operating System DOS Services* and *AIX Operating System DOS Services Reference*.

# dosdel

---

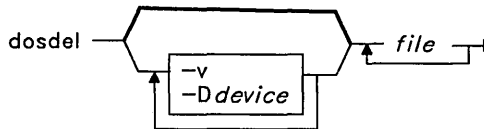
## dosdel

---

### Purpose

Deletes DOS files.

### Syntax



OL805108

### Description

The **dosdel** command deletes the DOS file specified by *file*. Use the **-v** flag to obtain format information about the disk.

File-naming conventions are those of DOS, with one exception. **doswrite** replaces the \ (backslash) character used to separate components of a DOS path name with the / (slash) because the backslash can have special meaning to AIX. **dosdel** converts lowercase characters in the *file1* name to uppercase before it checks the disk. Because all file names are assumed to be full (not relative) path names, you need not add the initial / (slash).

### Flags

- D device** Specifies a device or file system to use as the DOS disk. If you do not specify this flag the default device is **/dev/fd0**.
- v** Writes format information about the disk. Use primarily to verify the identify of a disk or file system as a DOS disk.

### Related Information

The following commands: “**dos**” on page 262, “**dosdir**” on page 267, “**dosread**” on page 269, and “**doswrite**” on page 271.

The **pcdos** subroutine in *AIX Operating System Technical Reference*.

---

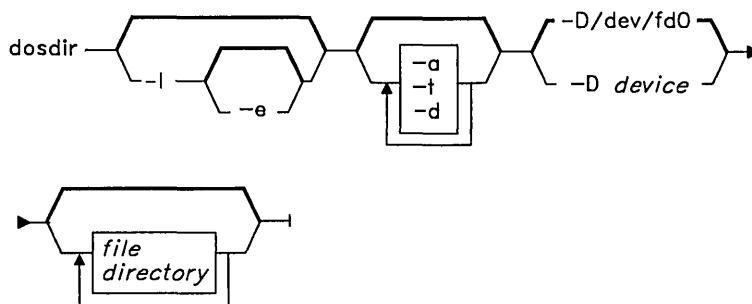
# dosdir

---

## Purpose

Lists the directory for DOS files.

## Syntax



OL805358

## Description

The **dosdir** command displays information about the specified DOS file or directory (the current directory by default). If you specify a directory without also specifying the **-d** flag, **dosdir** displays information about the files in that directory.

File-naming conventions are those of DOS, with one exception. **dosdir** replaces the \ (backslash) character used to separate components of a DOS path name with a / (slash) because the backslash can have special meaning to the AIX Operating System. **dosdir** converts lowercase characters in the file or directory name to uppercase before it checks the disk. Because all file names are assumed to be full (not relative) path names, you need not add the initial / (slash).

## Flags

- a**           Writes information about all files. This includes hidden and system files as well as the . (dot) and .. (dot dot) files.
- d**           Treats *file* as a file, even if it is a directory. If a directory is specified, information about the directory is listed rather than information about the files it contains.

## dosdir

---

- D [*device*] Specifies a device or file system to use as the DOS disk. If you do not specify this flag the default device is **/dev/fd0**.
- e Uses the **-l** flag to write the list of clusters allocated to the file.
- l Produces a long list that includes the creation date, size in bytes, and attributes. The size of a subdirectory is specified as 0 bytes. The attributes have the following meanings:
  - A Archive - the file has not been backed up since it was last modified.
  - D Directory - the file is a subdirectory, and is not included in the normal DOS directory search.
  - H Hidden - the file is not included in the normal DOS directory search.
  - R Read-only - the file cannot be modified.
  - S System - the file is a system file, and is not included in the normal DOS directory search.
- t Lists the entire directory tree starting at the named directory.
- v Writes information about the format of the disk.

## Related Information

The following commands: “**dosdel**” on page 266, “**dosread**” on page 269, and “**doswrite**” on page 271.

The **pcdos** subroutine in *AIX Operating System Technical Reference*.

---

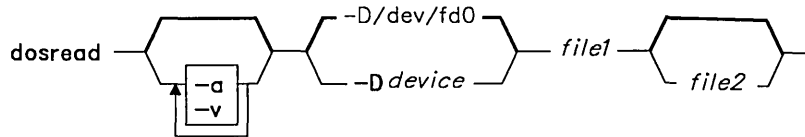
## dosread

---

### Purpose

Copies a DOS file.

### Syntax



OL805111

### Description

The **dosread** command copies the specified DOS *file1* to standard output or to the specified AIX *file2* (by default the root directory). Unless otherwise specified, **dosread** copies as many bytes as are specified in the directory entry for *file1*. This means, in particular, that copying directories does not work, since directories by convention have a record size of 0.

File-naming conventions are those of DOS, with one exception. **dosread** replaces the \ (backslash) character used to separate components of a DOS path name with a / (slash) because the backslash can have special meaning to the AIX Operating System. **dosread** converts lowercase characters in the *file1* name to uppercase before it checks the disk. Because all file names are assumed to be full (not relative) path names, you need not add the initial / (slash).

**Note:** Wild card characters (\* and ?) are not treated in a special way by this command (although they are by the shell). If, for example, you do not specify a file-name extension, the file name is matched as if you had specified a blank extension.

This command must be named **dosread**.

### Flags

- a Replaces the sequence CRLF (carriage return-line feed) with NL (new-line character) and interprets a **Ctrl-Z** (ASCII SUB) as the end-of-file character.

## dosread

---

- D device** Specifies the name of the DOS device or file system. The default *device* is **/dev/fd0**.
- Note:** This device must have the DOS-disk format.
- v** Writes information to the standard output about the format of the disk. Use this flag to verify that a device or file system is a DOS disk.

## Examples

1. To copy a text file from a DOS diskette to the AIX file system:

```
dosread -a chap1.doc chap1
```

This copies the DOS text file \CHAP1.DOC on **/dev/fd0** to the AIX file chap1 in the current directory.

2. To copy a nontext file from a fixed-disk DOS file system to the AIX file system:

```
dosread -D/dev/hd1 /survey/test.dta /u/fran/testdata
```

This copies the DOS data file \SURVEY\TEST.DTA on **/dev/hd1** to the AIX file /u/fran/testdata.

## Files

/dev/fd0

## Related Information

The following commands: “**dosdel**” on page 266, “**dosdir**” on page 267, and “**doswrite**” on page 271.

The **pcdos** subroutine in *AIX Operating System Technical Reference*.

---

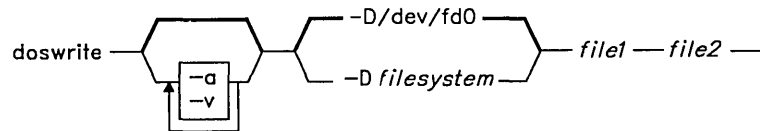
## doswrite

---

### Purpose

Copies AIX files to DOS files.

### Syntax



OL805112

### Description

The **doswrite** command copies the specified AIX *file1* to the specified DOS *file2*. If *file2* is a multi-component name (that is, if it contains /), each intervening component must exist as a directory and the last component (the named file), must not exist.

File-naming conventions are those of DOS, with one exception. **doswrite** replaces the \ (backslash) character used to separate components of a DOS path name with the / (slash) because the backslash can have special meaning to AIX. **doswrite** converts lowercase characters in the *file1* name to uppercase before it checks the disk. Because all file names are assumed to be full (not relative) path names, you need not add the initial / (slash).

**Note:** Wild card characters (\* and ?) are not treated in a special way by this command (although they are by the shell). If, for example, you do not specify a file-name extension, the file name is matched as if you had specified a blank extension.

This command must be named **doswrite**.

### Flags

- a Replaces NL (new-line character) characters with the sequence CR-LF (carriage return-linefeed). A Ctrl-Z is added to the output at end of file.
- D *filesystem* Specifies the name of the DOS device or file system. The default *device* is **/dev/fd0**.

**Note:** This device must have the DOS-disk format.



## doswrite

---

- v                   Writes information to the standard output about the format of the disk.  
Use this flag to verify that a device or file system is a DOS disk.

### Examples

1. To copy a text file from the AIX file system to a DOS diskette:

```
doswrite -a chap1 chap1.doc
```

This copies the AIX file chap1 in the current directory to the DOS text file \CHAP1.DOC on /dev/fd0.

2. To copy a nontext file from the AIX file system to a fixed-disk DOS file system:

```
doswrite -D/dev/hd1 /u/fran/testdata /survey/test.dta
```

This copies the AIX data file /u/fran/testdata to the DOS file \SURVEY\TEST.DTA on /dev/hd1.

### Files

/dev/fd0

### Related Information

The following commands: “**dosdir**” on page 267, “**dosread**” on page 269, and “**dosdel**” on page 266.

The **pcdos** subroutine in *AIX Operating System Technical Reference*.

## **dsipc**

---

### **Purpose**

Installs the Interprocess Communication key mapping in the kernel.

### **Syntax**

`dsipc —`

OL805461

### **Description**

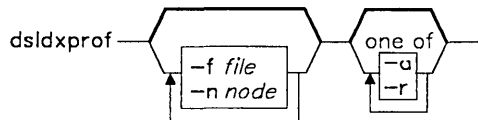
The **dsipc** command replaces all IPC key mapping currently in the kernel with new mapping from the profile data base. The **dsipc** command is usually called, at system startup time, by `/etc/rc.ds` to update the Distributed Services kernel. To use **dsipc** command from the command line, you must be a member of the system group or have superuser authority (see “**su**” on page 724).

### **Related Information**

“Using Distributed Services” in *Managing the AIX Operating System*.

**dsldxprof****dsldxprof****Purpose**

Loads translate information into the UID/GID translate profiles.

**Syntax**

OL805460

**Description**

The **dsldxprof** command loads translate information from a file into the UID/GID translate profiles. Each line in the file contains a row of translate information in the following format:

*Usr/Grp\_name U/G Local\_id Outbound\_id Inbound\_id Originating\_node*

This is the same format as the translate information from the **Network Users/Groups** table (for information on **Network Users/Groups**, see *Managing the AIX Operating System*). You must specify the *U/G*, *Local\_id*, and either the *Inbound\_id* or *Outbound\_id* fields. If you specify the *Inbound\_id* field the *Originating\_node* field must also be specified. A - (hyphen) is placed in unused fields as a placeholder.

**dsldxprof** reads a row of data from the file, validates the data, and loads the data into profiles. Translate rows are rejected due to improper syntax or incorrect values, or they may conflict with translate rows already in the profiles. A translate row is in conflict if there is an existing row in the profiles with a matching *U/G*, *Local\_id*, *Inbound\_id*, and *Originating\_node*, or if there is an existing row in the profiles with a matching *U/G*, *Local\_id*, and *Outbound\_id*, or both. If there is conflict, you are prompted to replace or reject the conflicting row. Rejected rows are written to standard error along with the information on why they are rejected.

To delete a translate row from the profiles, precede an identical row in the file with **##**.

To use **dsldxprof** command, you must be a member of the system group or have superuser authority (see “su” on page 724).

## | **Flags**

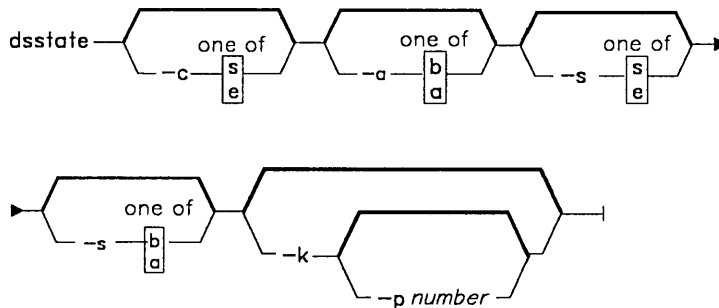
- |       **-a**           Places all rows that are in conflict into the profiles without prompting.
- |       **-f filename**   Reads translate information from *filename*.
- |       **-n nodename**   Updates translate profiles on the remote node *nodename*.
- |       **-r**            Rejects all conflicting rows without prompting.

## | **Related Information**

- |       “Using Distributed Services” in *Managing the AIX Operating System*.

**dsstate****dsstate****Purpose**

Sets the state of the Distributed Services kernel logic.

**Syntax**

OL805462

**Description**

The `dsstate` command changes the state of the Distributed Services kernel logic, including the number of kernel processes allocated for Distributed Services, whether incoming and outgoing remote requests are allowed, and where temporary storage takes place. Only members of the system group or users operating with superuser authority can use `dsstate` to change the state of the Distributed Services kernel logic (see “`su`” on page 724). Other users can use `dsstate` with no flags to write to the standard output the current state of the Distributed Services kernel logic.

**Flags**

- c s** Starts client sync, which forces all files for which this node is the client to be written directly to the server, preventing caching (temporary storage) of the file contents at the client. Starting client sync often affects the performance of file operations, and is used primarily for certain system startup and shutdown routines.
- c e** Ends client sync and allows some data to be stored at the local node.

- a b** Breaks all connections with remote nodes and blocks new requests for remote file services.
- a a** Allows requests from this client node for remote file services.
- s s** Starts server sync, which forces all files for which this node is the server to be written directly to the server, preventing caching (temporary storage) of the file contents at the client node. Starting server sync often affects the performance of file operations, and is used primarily for certain system startup and shutdown routines.
- s e** Ends server sync and allows some data to be stored at the client node.
- s b** Blocks all requests for file services from other nodes, including both new requests and requests for files already in use.
- s a** Allows this server to accept requests for file services from other nodes.
- k** Starts the Distributed Services kernel processes.
- p number** Sets the number of active Distributed Services kernel processes to *number*. If *number* is greater than the number of kernel processes allocated for Distributed Services, then those that are available are activated. If *number* is 0 or a negative value, the number of kernel processes is not changed.
- By adjusting the number of active Distributed Services kernel processes, the rate at which services are provided to remote nodes can be varied. Lowering the number of active Distributed Services kernel processes lowers remote use of this node's processor, leaving more system resources for local use.
- Note:** The Distributed Services kernel processes must have been started with a **-k** flag on either this **dsstate** command or an earlier **dsstate** command.

## Related Information

The **dsstate** system call in *AIX Operating System Technical Reference*.

"Using Distributed Services" in *Managing the AIX Operating System*.

## dsxlate

---

## dsxlate

---

### Purpose

Installs Distributed Services UID/GID translate tables into the kernel.

### Syntax

dsxlate —

OL805463

### Description

The **dsxlate** command installs Distributed Services UID/GID translate tables. **dsxlate** is usually called, at system startup, by **/etc/rc.ds** to update the kernel. To use **dsstate** command from the command line, you must be a member of the system group or have superuser authority (see “**su**” on page 724).

**dsxlate** ensures that the Distributed Services kernel tables reflect the current profiles. All existing Distributed Services kernel information is discarded.

### Related Information

The following commands: “**ipctable**” on page 414.1, “**ndtable**” on page 506.1, and “**ugtable**” on page 784.

The **loadtbl** system call in *AIX Operating System Technical Reference*.

“Using Distributed Services” in *Managing the AIX Operating System*.

---

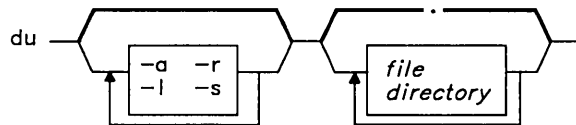
# du

---

## Purpose

Summarizes disk usage.

## Syntax



OL805113

## Description

The **du** command gives the number of blocks in all files and (recursively), directories within each specified *directory*. By specifying the **-a** flag, you can also have **du** report the number of blocks in individual files. The block count includes the indirect blocks of each file and is in units of 512 bytes, independent of the cluster size used by the system. If you provide no *file* or *directory* name, **du** uses the current directory.

**Note:** If you do not specify the **-a** flag, **du** does not report on any *files*.

If there are too many distinct linked files, **du** counts the excess files more than once.

Block counts are based only on file size; therefore, unallocated blocks are not accounted for in the block counts reported.

## Flags

- a** Displays disk use for each file.
- l** Allocates blocks in files with multiple links evenly among the links. By default, a file with two or more links is counted only once.
- r** Indicates inaccessible files and directories.
- s** Displays only the grand total (for each of the specified files or directories given).



## Examples

1. To summarize the disk usage of a directory tree and each of its subtrees:

```
du /u/fran
```

For `/u/fran` and each of its subdirectories, this displays the number of disk blocks that the files in the tree beneath it contain.

2. To display the disk usage of each file:

```
du -a /u/fran
```

This displays the number of disk blocks contained in each file and subdirectory of `/u/fran`. The number beside a directory is the disk usage of that directory tree. The number beside a regular file is the disk usage of that file alone.

3. To display only the total disk usage of a directory tree:

```
du -rs /u/fran
```

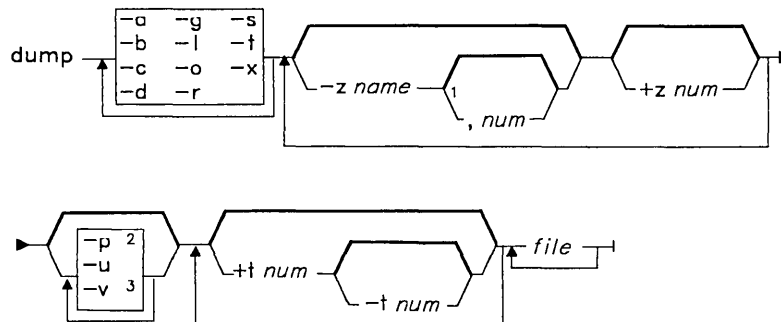
This displays only the sum total disk usage of `/u/fran` and the files it contains (`-s`). The `-r` flag tells **du** to display an error message if it cannot read a file or directory.

# dump

## Purpose

Dumps selected parts of an object file.

## Syntax



<sup>1</sup> Do not put a space between these items.

<sup>2</sup> Use `-p` only with `-a`, or `-o`.

<sup>3</sup> Do not use `-v` with `-s` or `-o`.

OL805404

## Description

The **dump** command dumps selected parts of the specified *file*. **dump** accepts object files, archive object files, and executable files (with the `-x` flag). It writes information in character, hexadecimal, octal, or decimal representation, as appropriate to format the information in a meaningful way.

## Flags

You must use at least one of the following flags:

- a** Dumps the archive header of each member of each specified archive.
- b** Dumps the shared library key.
- c** Dumps the string table.

## dump

---

- d** Dumps the contents of the data section.
- g** Dumps the global symbols in the archive symbol table.
- l** Dumps line number information.
- o** Dumps each optional header.
- r** Dumps relocation information.
- s** Dumps the contents of the object file section.
- t** Dumps symbol table entries.
- x** Dumps the object module extended header from executable files. The extended header contains the table of shared libraries that the program uses.

The following optional flags are also available:

- p** Does not print the headers.
- tnum** Dumps only the index symbol table entry specified with *num*. Use **-t** with the **+t** flag to specify a range of symbol table entries.
- +tnum** Dumps the symbol table entry in the range that ends with *num*. The range starts at either the first symbol table entry or at the entry specified by **-t**.
- u** Underline the name of the *file*.
- v** Dumps the information in symbolic representation rather the numeric. You can use this with any of the above flags except **-s** or **-o**.
- zname[,num]** Dumps line number entries for *name* function or a range of line number entries that starts at the specified number. You can use a blank to replace the comma that separates *name* and *num* if the entire argument is quoted.
- +znum** Dumps all line numbers up to *num*.

## Related Information

The following commands: “**ar**” on page 58, “**nm**” on page 521, “**shlib**” on page 660, and “**size**” on page 665.

The **a.out** and **ar** files in *AIX Operating System Technical Reference*.

---

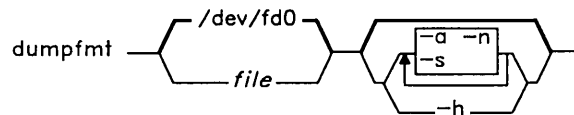
# dumpfmt

---

## Purpose

Formats the VRM dump file.

## Syntax



OL805109

## Description

The **dumpfmt** command formats a file containing VRM dump structures. If you do not specify a *file* name, the system reads data from **/dev/fd0**.

By default, **dumpfmt** is an interactive utility program. To see the list of commands available for selecting a specific structure to format, enter a **?**. To quit, enter **q**.

## Flags

- a** Batches the output and formats the entire diskette.
- h** Includes a Dump Data Header. This header contains general information about data on the dump diskette: the module name of the component, the data address of the module containing the component, and the offset address within the module of the component.
- n** Does not display a prompt when the screen fills with data during interactive output.
- s** Limits the output of each structure to a maximum size of 32 bytes.

## Related Information

The discussion of **dumpfmt** in *AIX Operating System Programming Tools and Interfaces*.

# echo

---

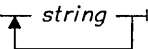
## echo

---

### Purpose

Writes its arguments to standard output.

### Syntax

echo  *string*

OL805115

### Description

The **echo** command writes its arguments to standard output. *strings* are separated by blanks and a new-line character follows the last *string*. Use **echo** to produce diagnostic messages in command files and to send data into a pipe.

The **echo** command recognizes the following escape conventions:

- \b**     Display a backspace character.
- \c**     Suppress the new-line character.
- \f**     Display a form-feed character.
- \n**     Display a new-line character.
- \r**     Display a carriage return character.
- \t**     Display a tab character.
- \\**     Display a backslash character.
- \num**   Display an 8-bit character. whose ASCII value is the 1-, 2-or 3-digit octal number *num*. The first digit of *num* must be a zero.

### Examples

1. To write a message to standard output:  
    echo Please insert diskette . . .

2. To display a message containing special characters:

```
echo "\n\n\nI'm at lunch.\nI'll be back at 1:00."
```

This skips three lines and displays the message:

```
I'm at lunch.  
I'll be back at 1:00.
```

**Note:** You must quote the message if it contains escape sequences like `\n`. Otherwise, the shell treats the `\` specially. See page 641 for details about quoting.

3. To use **echo** with pattern-matching characters:

```
echo The back-up files are: *.bak
```

This displays the message `The back-up files are:` followed by the file names in the current directory ending with `.bak`.

4. To add a single line of text to a file:

```
echo Remember to set the shell search path to $PATH. >>notes
```

This adds the message to the end of the file `notes` after the shell substitutes the value of the shell variable **PATH**.

5. To write a message to the standard error output:

```
echo Error: file already exists. >&2
```

Use this in shell procedures to write error messages. If the `>&2` is omitted, then the message is written to the standard output. For details about this type of file redirection, see “Input and Output Redirection Using File Descriptors” on page 651.

## Related Information

The following command: “**sh**” on page 637.

# ed

---

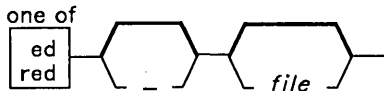
# ed

---

## Purpose

Edits text by line.

## Syntax



OL805182

## Description

The **ed** command is a line editing program that works on only one file at a time by copying it into a temporary file buffer and making changes to that copy. **ed** does not alter the file itself until you use the **w**rite subcommand. You can specify on the command line the *file* you want to edit, or you can use the **e**dit subcommands. If you specify a - (minus) on the command line, **ed** does not display character counts with the **e**, **r**, or **w** subcommands, diagnostic messages with the **e** or **q** subcommands, or the **!** prompt after a *!AIX-cmd*.

When **ed** reads a new file into the buffer, the contents of that file replaces the buffer's previous contents, if any.

There is also a restricted version of **ed**, the **red** command, for use with the restricted shell (see "**sh**" on page 637). With **red**, you can edit only files that reside in the current directory, or in the */tmp* directory, and you cannot use the *!AIX-cmd* subcommand (see page 290).

An **ed** subcommand consists of zero, one, or two *addresses*, followed by a single-character subcommand, possibly followed by parameters to that subcommand. These addresses specify one or more lines in the buffer. Because every subcommand has default addresses, you frequently do not need to specify addresses.

The **ed** program operates in one of two modes, *command mode* and *text mode*. In command mode, **ed** recognizes and executes subcommands. In text mode, **ed** adds text to the file buffer but does not recognize subcommands. To leave text mode, enter a . (dot) alone at the beginning of a line.

## Pattern Matching

The **ed** command supports a limited form of *special pattern-matching characters* that you can use as *regular expressions (REs)* to construct *pattern strings*. You can use these patterns in addresses to specify lines and in some subcommands to specify portions of a line.

### Regular Expressions (REs)

The following REs match a *single* character:

- char* An ordinary *character* (one other than one of the special pattern-matching characters), matches itself.
- A period (.) matches any single character except for the new-line character.
- [*string*] A *string* enclosed in square brackets ([ ]) matches *any one* character in that string. Certain pattern-matching characters have special meanings within square brackets:
- ^ If the first character of *string* is a circumflex, then the RE ([^*string*]) matches any character *except* the characters in *string* and the new-line character. A ^ has this special meaning *only* if it occurs first in the string.
  - You can use a minus (-) to indicate a range of consecutive ASCII characters according to the current collating sequence. For example, [a-f] might be equivalent to [abcdef] or [aAbBcCdDeEfF] or [aâãbcdeêëf]. The collating sequence is defined by the environment variable **NLCTAB** or **NLFILE**. See *Managing the AIX Operating System* for more information. A collating sequence may define “equivalence classes” for characters. For example, if three characters—e, é, and è—are equivalent, the following expressions identify the same sequence of characters:
 

```
[a-e]
[a-è]
```

 The minus character loses its special meaning if it occurs first ([-*string*]), if it immediately follows an initial circumflex ([^*string*]), or if it appears last ([*string*-]) in the string.
  - ] When the right square bracket (]) is the first character in the string ([*string*]) or when it immediately follows an initial circumflex ([^]*string*]), it is treated as a part of the string rather than as the string terminator.
- \*sym* A \ (backslash) followed by a special pattern-matching character matches the special character itself (as a literal character). These special pattern-matching characters are:



- . \* [ \ Always special *except* when they appear within square brackets ([ ]).
- ^ Special at the *beginning* of an entire pattern or when it immediately follows the left bracket of a pair of brackets ([^ . . . ]).
- \$ Special at the *end* of an entire pattern.

In addition, the character used to delimit an entire pattern is special for that pattern. (For example, see how slash (/) is used in the **g** subcommand on page 286.)

## Forming Patterns

The following rules describe how to form patterns from REs:

1. An RE that consists of a single, ordinary character matches that same character in a string.
2. An RE followed by an asterisk (\*) matches zero or more occurrences of the character that the RE matches. For example, the following pattern:

```
ab*cd
```

matches each of the following strings:

```
acd  
abcd  
abccd  
abbbcd
```

but not the following string:

```
abd
```

If there is any choice, the longest matching leftmost string is chosen. For example, given the following string:

```
122333444
```

the pattern `.*` matches 122333444, the pattern `.*3` matches 122333, and the pattern `.*2` matches 122.

3. An RE followed by:

- `\{m\}` Matches *exactly m* occurrences of the character matched by the RE.
- `\{m,\}` Matches *at least m* occurrences of the character matched by the RE.
- `\{m,n\}` Matches *any number* of occurrences of the character matched by the RE *from m to n inclusive*.

*m* and *n* must be integers from 0 to 255, inclusive. Whenever a choice exists, this pattern matches as many occurrences as possible.

4. You can combine REs into patterns that match strings containing that same sequence of characters. For example, `AB\*CD` matches the string `AB*CD` and `[A-Za-z]\*[0-9]\*` matches any string that contains any combination of alphabetic characters (including none), followed by any combination of numerals (including none).
5. The character sequence `\(pattern\)` marks a *subpattern* that matches the same string it would match if it were not enclosed.
6. The characters `\num` match the same string of characters that a subpattern matched earlier in the pattern (see the preceding discussion of item 5). `num` is a digit. The pattern `\num` matches the string matched by the `numth` subpattern, counting from left to right. For example, the following pattern:

```
\(A\)\(B\)C\2\1
```

matches the string `ABCBA`. You can nest subpatterns.

## Restricting What Patterns Match

A pattern can be restricted to match only the first segment of a line, the final segment, or both:

1. A `^` (circumflex) at the beginning of a pattern causes the pattern to match only a string that begins in the first character position on a line.
2. A `$` (dollar sign) at the end of a pattern causes that pattern to match only a string that ends with the last character (not including the new-line character) on a line.
3. The construction `^pattern$` restricts the pattern to matching only an entire line.

In addition, the null pattern (that is, `//`) duplicates the previous pattern.

## Addressing

The *current line*, usually the last line affected by a command, is the point of reference in the buffer. `ed` always has a current line. This is the default address for several `ed` commands. (See “Subcommands” on page 285 to find out how each subcommand affects the current line.)

There are three types of `ed` addresses: line number addresses, addresses relative to the current line, and pattern addresses. Following are guidelines for constructing addresses:

1. `.` (dot) addresses the current line.
2. `$` (dollar sign) addresses the last line of the buffer.
3. `n` addresses the *n*th line of the buffer.
4. `'x` addresses the line marked with a lowercase ASCII letter, *x*, by the `k` subcommand (see page 287).

5. */pattern/* (a pattern enclosed in slashes) addresses the next line contains a matching string. The search begins with the line after the current line and stops when it finds a match for the pattern. If necessary, the search moves to the end of the buffer, wraps around to the beginning of the buffer, and continues until it either finds a match or returns to the current line.
6. *?pattern?* (a pattern enclosed in question marks) addresses the previous line that contains a match for the pattern. The *?pattern?* construct, like */pattern/*, can search the entire buffer, but it does so in the opposite direction.
7. An address followed by *+n* or *-n* (a plus sign or a minus sign followed by a decimal number) specifies an address plus or minus the indicated number of lines. (The *+* sign is optional.)
8. An address that begins with *+* or *-* specifies a line relative to the current line. For example, *-5* is the equivalent of *.-5* (five lines above the current line).
9. An address that ends with *-* or *+* specifies the line immediately before (*-*) or immediately after (*+*) the addressed line. Used alone, the *-* character addresses the line immediately before the current line. The *+* character addresses the line immediately after the current line; however, the *+* character is optional. The *+* and *-* characters have a cumulative effect; for example, the address *--* addresses the line two lines above the current line.
10. For convenience, a *,* (comma) stands for the address pair *1,\$* (first line through last line) and a *;* (semicolon) stands for the pair *.,\$* (current line through last line).

Commands that do not accept addresses regard the presence of an address as an error. Commands that do accept addresses can use either given or default addresses. When given more addresses than it accepts, a command uses the last (rightmost) one(s).

In most cases, commas (*,*) separate addresses (for example *2,8*). Semicolons (*;*) also can separate addresses. A semicolon between addresses causes **ed** to set the current line to the first address and then calculate the second address (for example, to set the starting line for a search based on rules 5 and 6 above). In a pair of addresses, the first must be numerically smaller than the second.

For many purposes, you may prefer to use a different editor that has different features:

“**edit**” on page 292, a simple line editor for novice or casual users

“**sed**” on page 629, a stream editor often used for writing programs

“**ex**” on page 312, an extended (line) editor with numerous interactive subcommand features

“**vi**, **vedit**, **view**” on page 832, a visual (screen) editor that also accesses **ex** line editing features while letting you view the text.

The following is a list of **ed** size limitations:

- 64 characters per file name.
- 512 characters per line (although there is currently a system-imposed limit of 255 characters per line entered from the keyboard).
- 256 characters per global subcommand list.
- 128K characters buffer size. (Note that the buffer not only contains the original file but also editing information. Each line occupies one word in the buffer.)

In addition, the maximum number of lines permitted also depends on the amount of memory available to you. The maximum file size depends on the amount of physical data storage (disk or tape drive) available or on the maximum number of lines permitted in user memory.

## Subcommands

In most cases, only one **ed** subcommand can be entered on a line. The exceptions to this rule are the **p** and **l** subcommands, which can be added to any **ed** command except **e**, **f**, **r**, or **w**.

The **e**, **f**, **r**, and **w** subcommands accept file names as parameters. The **ed** program stores the last file name used with a subcommand as a default file name. The next **e**, **f**, **r**, or **w** given without a file name uses the default file name.

The **ed** program responds to an error condition with one of two messages: **?** (question mark) or **?file**.

When **ed** receives an INTERRUPT signal (**Alt-Pause**), it displays a **?** and returns to command mode.

When it reads a file, **ed** discards ASCII NULL characters and all characters after the last new-line character. **ed** cannot edit a file that contains characters not in the ASCII set (for example, an **a.out** file with bit 8 set on).

**Note:** In the following list of **ed** subcommands, default addresses are shown in parentheses. (Do not key in the parentheses.) The address **.** (period) refers to the current line.

**(.)a**  
**< text >**

**.**

The **append** subcommand adds text to the buffer after the addressed line. The **a** subcommand sets the current line to the last inserted line, or, if no lines were inserted, to the addressed line. Address **0** causes the **a** subcommand to add text at the beginning of the buffer.

- (.)**c**  
< *text* >  
.
- The **change** subcommand deletes the addressed lines, then replaces them with new input. The **c** command sets the current line to the last new line of input, or, if there were none, to the first line that was not deleted.
- (.,.)**d**
- The **delete** subcommand removes the addressed lines from the buffer. The line after the last line deleted becomes the current line. If the deleted lines were originally at the end of the buffer, the new last line becomes the current line.
- e** *file*
- The **edit** subcommand first deletes any contents from the buffer, then loads another file into the buffer, sets the current line to the last line of the buffer, and displays the number of characters read in to the buffer. If the buffer has been changed since its contents were last saved (with the **w** subcommand), **e** displays **?** before it clears the buffer.
- The **e** subcommand stores *file* as the default file name to be used, if necessary, by subsequent **e**, **r**, or **w** subcommands. (See the **f** subcommand.)
- When the **!** character replaces *file*, **e** takes the rest of the line as a AIX shell (**sh**) command and reads the command output. The **e** subcommand does not store the name of the shell command as a default file name.
- E** *file*
- The **Edit** subcommand works like **e**, with one exception: **E** does not check for changes made to the buffer since the last **w** subcommand.
- f** [*file*]
- The **file name** subcommand changes the default file name (the stored name of the last file used) to *file*, if *file* is given. If *file* is not given, the **f** subcommand prints the default file name.
- (1,?)**g**/*pattern/subcmd-list*
- The **global** subcommand first marks every line that matches the pattern. Then, for each marked line, this subcommand sets the current line to that line and executes *subcmd-list*. A single subcommand, or the first subcommand of a list, should appear on the same line with the **g** subcommand; subsequent subcommands should appear on separate lines. Except for the last line, each of these lines should end with a **\**.
- The *subcmd-list* can include the **a**, **i**, and **c** subcommands and their input. If the last command in *subcmd-list* would normally be the **.** (dot) that ends input mode, the **.** (dot) is optional. If there is no *subcmd-list*, **ed** displays the current line. The *subcmd-list* cannot include the **g**, **G**, **v**, or **V** subcommands.

**Note:** The **g** subcommand is similar to the **v** subcommand, which executes *subcmd-list* for every line that does not contain a match for the pattern.

(1,?)**G**/*pattern*/

The interactive **Global** subcommand first marks every line that matches the pattern, then displays the first marked line, sets the current line to that line, and waits for a subcommand. **G** accepts any but the following **ed** subcommands: **a**, **c**, **i**, **g**, **G**, **v**, and **V**. After the subcommand finishes, **G** displays the next marked line, and so on. **G** takes a new-line character as a null subcommand. An **:&** causes **G** to execute the previous subcommand again, if there was one. Note that subcommands executed within the **G** subcommand can address and change any lines in the buffer. The **G** subcommand can be terminated by pressing **INTERRUPT (Alt-Pause)**.

**h**

The **help** subcommand gives a short explanation (help message) for the most recent **?** diagnostic or error message.

**H**

The **Help** subcommand causes **ed** to display the help messages for all subsequent **?** diagnostics. **H** also explains the previous **?** if there was one. **H** alternately turns this mode on and off; it is initially off.

(.)**i**  
<*text*>  
.

The **insert** subcommand inserts text before the addressed line and sets the current line to the last inserted line. If there no lines are inserted, **i** sets the current line to the addressed line. This subcommand differs from the **a** subcommand only in the placement of the input text. Address **0** is not legal for this subcommand.

(.,+1)**j**

The **join** subcommand joins contiguous lines by removing the intervening new-line characters. If given only one address, **j** does nothing. (For splitting lines, see the **s** subcommand.)

(.)**kx**

The **mark** subcommand marks the addressed line with name *x*, which must be a lowercase ASCII letter. The address **'x** (single quotation mark before the marking character) then addresses this line. The **k** subcommand does not change the current line.

(.,)**l**

The **list** subcommand displays the addressed line(s). The **l** subcommand wraps long lines and, unlike the **p** subcommand, represents non-printing characters, either with mnemonic overstrikes or in octal notation. An **l** subcommand may be appended to any **ed** subcommand except: **e**, **f**, **r**, or **w**.

- (.,.)**m***a* The **move** subcommand repositions the addressed line(s). The first moved line follows the line addressed by *a*. Address 0 for *a* causes **m** to move the addressed line(s) to the beginning of the file. Address *a* cannot be one of the lines to be moved. The **m** subcommand sets the current line to the last moved line.
- (.,.)**n** The **number** subcommand displays the addressed lines, each preceded by its line number and a tab character (displayed as blank spaces); **n** leaves the current line at the last line displayed. An **n** subcommand may be appended to any **ed** subcommand except **e**, **f**, **r**, or **w**.
- (.,.)**p** The **print** subcommand displays the addressed line(s) and sets the current line set to the last line displayed. A **p** subcommand may be appended to any **ed** subcommand except: **e**, **f**, **r**, or **w**. For example, the subcommand **dp** deletes the current line and displays the new current line.
- P** The **P** subcommand turns on or off the **ed** prompt string \* (asterisk). Initially, **P** is off.
- q** The **quit** subcommand exits the **ed** program. Before ending the program **q** checks to determine whether the buffer has been written to a file since the last time it was changed. If not, **q** displays the ? message.
- Q** The **Quit** subcommand exits the **ed** program without checking for changes to the buffer since the last **w** subcommand (compare with the **q** subcommand).
- (?)**r** *file* The **read** subcommand reads a file into the buffer after the addressed line; **r** does not delete the previous contents of the buffer. When entered without *file*, **r** reads the default file, if any, into the buffer (see **e** and **f** subcommands). **r** does not change the default file name. Address 0 causes **r** to read a file in at the beginning of the buffer. After it reads a file successfully, **r**, displays the number of characters read into the buffer and sets the current line to the last line read. If ! (exclamation point) replaces *file* in a **r** subcommand, **r** takes the rest of the line as a AIX shell (**sh**) command whose output is to be read. The **r** subcommand does not store the names of shell commands as default file names.
- (.,.)**s**/*pattern*/*replacement*/  
(.,.)**s**/*pattern*/*replacement*/**g** The **substitute** subcommand searches each addressed line for a string that matches the *pattern* and then replaces the string with the specified *replacement* string. Without the global

indicator (**g**), **s** replaces only the first matching string on each addressed line. With the **g** indicator, **s** replaces every occurrence of the matching string on each addressed line. If **s** does not find a match for the pattern, it returns the error message **?**. Any character except a space or a new-line character can separate (delimit) the pattern and *replacement*. The **s** subcommand sets the current line to the last line changed.

An ampersand (&) in the *replacement* string is a special symbol that has the same value as the *pattern* string. So, for example, the subcommand **s/are/&n't/** has the same effect as the subcommand **s/are/aren't/** and replaces **are** with **aren't** on the current line. A backslash before the ampersand (\&) removes this special meaning of & in *replacement*.

A subpattern is part of a pattern enclosed by the strings **(** and **)**; the pattern works as if the enclosing characters were not present. In *replacement*, the characters **\n** refer to strings that match subpatterns; **n**, a decimal number, refers to the *n*th subpattern, counting from the left. (for example, **s/(t)(h)\(e)/t1\2ose** replaces **the** with **those** if there is a match for the pattern **the** on the current line). Whether subpatterns are nested or in a series, **\n** refers to the *n*th occurrence, counting from the left, of the delimiting characters, **(**).

The **%** (percent sign) character, when used by itself as *replacement*, causes **s** to use the previous *replacement* again. The **%** character does not have this special meaning if it is part of a longer *replacement* or if it is preceded by a **\**.

Lines may be split by substituting new-line characters into them. In *replacement*, the sequence **\Enter** quotes the new-line character (not displayed) and moves the cursor to the next line for the remainder of the string. New-lines cannot be substituted as part of a **g** or **v** subcommand list.

(.,)ta

The transfer subcommand inserts a copy of the addressed lines after address *a*. The **t** subcommand accepts address 0 (for inserting lines at the beginning of the buffer). The **t** subcommand sets the current line to the last line copied.

u

The undo subcommand restores the buffer to the state it was in before it was last modified by an **ed** subcommand. The commands that **u** can undo are: **a**, **c**, **d**, **g**, **G**, **i**, **j**, **m**, **r**, **s**, **t**, **v**, and **V**.



- (1,?)**v**/*pattern/subcmd-list* The **v** subcommand executes the subcommands in *subcmd-list* for each line that does not contain a match for the pattern.  
**Note:** The **v** subcommand is a complement for the global subcommand **g**, which executes *subcmd-list* for every line that does contain a match for the pattern.
- (1,\$)**V**/*pattern//* The **V** subcommand first marks every line that does not match the pattern, then displays the first marked line, sets the current line to that line, and waits for a subcommand.  
**Note:** The **V** subcommand complements the **G** subcommand, which marks the lines that do match the pattern.
- (1,?)**w** *file* The **w** write subcommand copies the addressed lines from the buffer to the file named in *file*. If the file does not exist, the **w** subcommand creates it with permission code 666 (read and write permission for everyone), unless the **umask** setting specifies another file creation mode. (For information about file permissions, see “**umask**” on page 784 and “**chmod**” on page 128.) The **w** subcommand does not change the default file name (unless *file* is the first file name used since you started **ed**). If you do not provide a file name, **ed** uses the default file name, if any (see the **e** and **f** subcommands). The **w** subcommand does not change the current line.  
  
If **ed** successfully writes the file, it displays the number of characters written. When **!** replaces *file*, **ed** takes the rest of the line as a AIX shell (**sh**) command whose output is to be read; **w** does not save shell command names as default file names.  
  
**Note:** 0 is not a legal address for the **w** subcommand. Therefore, it is not possible to create an empty file with **ed**.
- (**\$**)=  
Without an address, the = (equal sign) subcommand displays the current line number. With the address **\$**, = displays the number of the last line in the buffer. The = subcommand does not change the current line and cannot be included in a **g** or **v** subcommand list.
- !AIX-cmd** The **!** (exclamation point) subcommand allows AIX commands to be run from within **ed**. Anything following **!** on an **ed** subcommand line is interpreted as an AIX command. Within the text of that command string, **ed** replaces the unescaped character **%** with the current file name, if there is one.  
  
When used as the first character of a shell command (after the **!** that runs a subshell) **ed** replaces the **!** character with the

previous AIX command; for example, the command `!!` repeats the previous AIX command. If the AIX command interpreter (the `sh` command), expands the command string, `ed` echoes the expanded line. The `!` subcommand does not change the current line.

*num*  
`+ num`  
`- num`

`ed` interprets a number alone on a line as an address and displays the addressed line. Addresses can be absolute (line numbers or `$`) or relative to the current line (`+num` or `- num`). Entering a new-line character (a blank line) is equivalent to `+1p` and is useful for stepping forward through the buffer one line at a time.

## Files

`/tmp/e#` Temporary file; # is the process number.  
`ed.hup` Work is saved here if the terminal hangs up while `ed` is running.

## Related Information

The following commands: “`grep`” on page 381, “`sed`” on page 629, “`sh`” on page 637, “`stty`” on page 717, and “`regcmp`” on page 595.

The `regexp` system call in *AIX Operating System Technical Reference*.

The `environment` miscellaneous facility in *Text Formatting Guide*.

The discussion and examples of `ed` in *Using the AIX Operating System*.

The “Overview of International Character Support” in *Managing the AIX Operating System*.

# edit

---

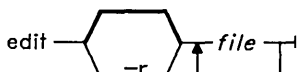
## edit

---

### Purpose

Provides a simple line editor for the new user.

### Syntax



OL805329

### Description

**Warning:** The `edit` command does not support international characters. If you use this command to edit a file that contains extended characters, you can lose data.

The `edit` command provides a line editor designed for beginning users. It is a simplified version of the `ex` command (see “`ex`” on page 312). To edit the contents of a file, enter:

```
edit file
```

If *file* is the name of an existing file, `edit` copies it to a buffer and displays the number of lines and characters in it. Then it displays a colon prompt (`:`) to show that it is ready to read subcommands from standard input. If *file* does not already exist, `edit` tells you this, but still stores the name as the current file name. You can give more than one *file* name, in which case `edit` copies the first file into its buffer and stores the remaining file names in an **argument list** for later use.

The `edit` command operates in one of two modes: **command mode** and **text entry mode**. In command mode, `edit` displays the colon prompt to show you that it is ready to accept `edit` subcommands. In text entry mode, `edit` places all input into its editing buffer. The general format of an `edit` subcommand is as follows:

```
[addr]subcommand [parameters] [count]
```

If you do not specify an *address*, `edit` works on the **current line**. If you add a numeric *count* to most subcommands, `edit` works on the specified number of lines.

For most subcommands, the last line affected becomes the new current line. That means, for example, that after `edit` reads a file into its buffer, the last line in the file becomes the current line. *addr* can be a line number or a *pattern* to be matched or, in some cases, a range of line numbers or *patterns*. To specify a range, separate two line numbers or

*patterns* with a comma or a semicolon (for example, 1,5 or 1;5). In a range, the second address must refer to a line that follows the first addressed line in the range.

## Addressing Lines Within a File

The simplest way to address a line within a file is to use its line number. But this can be unreliable because line numbers change when you insert and delete lines. **edit** provides a way to search through the buffer for strings. Given the following address:

```
/pattern/
```

**edit** searches forward for *pattern*, while given:

```
?pattern?
```

it searches backwards for *pattern*. If a forward search reaches the end of the buffer without finding *pattern*, it continues the search at the beginning of the file until it reaches the current line. A backwards search does just the reverse.

The following characters have special meanings in these search patterns:

^           Matches the beginning of a line.

\$           Matches the end of a line.

Thus, you can use `/^pattern/` to search for patterns at the beginning of a line, and `/pattern$/` to search for patterns at the end of the line.

The current line has a symbolic name, dot (`.`), and the last line in the buffer has a symbolic name, dollar sign (`$`), that you can use in addresses. This is useful when working with a range of lines. For example,

```
.,$print
```

displays all lines from the current line to the last line in the buffer. Arithmetic with line references is also possible, so that `$-5` refers to the fifth line from the last and `+.20` refers to the line 20 lines past the current line. You can also use the `=` (equal) command to find out the line number of the current line or the last line, as follows:

```
. =  
$ =
```

To view the next line in the buffer, press the **Enter** key. Press **Ctrl-D** to display the next half-screen of lines.

**Note:** Do not confuse the meaning of `$` in text patterns (end of line) with its meaning in addresses (last line).

## Using the ex Command

As you become more experienced with using an editor, you may still find that **edit** meets your needs. If you become interested in using **ex**, you will find that it builds on the commands you are already familiar with from using **edit**.

The **edit** subcommands work the same way in **ex**, but the editing environment is somewhat different. You should be aware of the differences that exist between the two editors. In **edit**, only the characters **^**, **\$**, and **\** have special meanings as pattern-matching characters. Several additional characters have special meanings in **ex**, as described under “**ex**” on page 312.

Another feature of the **edit** environment prevents you from accidentally entering two alternative modes of editing, *open mode* and *visual mode*, in which the editor behaves differently from normal command mode. See “**vi, vedit, view**” on page 832 for a full discussion of visual mode.

## Flag

**-r** Recovers *file* after an editor or system crash.

## Subcommands

You can enter most **edit** subcommands as either a complete word or an abbreviation. In the following list, a subcommand abbreviation appears in parentheses. Unless noted otherwise, all subcommands work by default on the current line. **edit** recognizes and interprets the following subcommands when it displays the colon prompt:

**[addr]append (a)**  
*text*

.

Reads the input *text* into the file being edited, placing the text after the line at the specified *address*. If you specify address 0, **edit** places the text at the beginning of the buffer.

**[addr1[,addr2]]change (c)**  
*text*

.

Replaces the specified line or lines with the input *text*. If any lines are input, the last input line becomes the new current line.

**[addr1[,addr2]]delete [buffer] (d)**

Removes the specified line or lines from the editing buffer. The line following the last deleted line becomes the current line. If you specify a *buffer* by giving a letter from a to Z, **edit** saves the specified lines in that buffer or, if the letter is uppercase, appends the lines to that buffer.

- edit file** (e) Begins an editing session on a new file. The editor first checks to see if the buffer has been modified (*edited*) since the last **write** subcommand. If it has, **edit** issues a warning and cancels the **edit** subcommand. Otherwise, it deletes the complete contents of the editor buffer, makes the named file the current file, and displays the new file name. After insuring that this file can be edited, it reads the file into its buffer. If **edit** reads the file without error, it displays the number of lines and characters that it read. The last line read becomes the new current line.
- file** (f) Displays the current file name along with the following information about it:
- Whether it has been modified since the last **write**.
  - What the current line is.
  - How many lines are in the buffer.
  - What percentage of the way through the buffer the current line is.
- file file** Changes the name of the current file to *file*. **edit** considers this file *not edited*.
- [addr1[,addr2]]global/pattern/cmds** (g) Marks each of the specified lines that matches the *pattern*. Then **edit** carries out the specified subcommands (*cmds*) on each marked line.
- A single *cmd* or the first *cmd* in a subcommand list appears on same line as **global**. The remaining *cmds* must appear on separate lines, where each line (except the last) ends with a \ (backslash). The default subcommand is **print**.
- The list can include the **append**, **insert**, and **change** subcommands and their associated input. In this case, if the ending period comes on the last line of the command list, you may omit it. The **undo** subcommand and the **global** subcommand itself, however, may not appear in the command list.
- [addr]insert** (i)  
*text*  
.
- Places the given text before the specified line. The last line input becomes the current line. Otherwise, the current line does not change.
- [addr1[,addr2]]move addr3** (m) Repositions the specified line or lines to follow *addr3*. The first of the moved lines becomes the current line.
- next** (n) Copies the next file in the command line argument list to the buffer for editing.
- [addr1[,addr2]]number** (nu) Displays each specified line or lines preceded by its buffer line number. The last line displayed becomes the current line.

**preserve**

Saves the current editor buffer as though the system had just crashed. Use this command when a **write** subcommand has resulted in an error, and you do not know how to save your work.

**[addr1[,addr2]]print (p)**

Displays the specified line or lines. The last line displayed becomes the current line.

**[addr]put buffer (pu)**

Retrieves the contents of the specified buffer and places it after *addr*. If you do not specify a buffer, **edit** restores the last deleted or yanked text. Thus you can use this subcommand together with **delete** to move lines or with **yank** to duplicate lines between files.

**quit (q)****quit! (q!)**

Ends the editing session.

**Note:** The **quit** command does *not* write the editor buffer to a file. However, if you have modified the contents of the buffer since the last **write**, **edit** displays a warning message and does not end the session. In this case, either use the **quit!** subcommand to discard the buffer or **write** the buffer and then **quit**.

**recover file**

Recovers *file* from the system save area. Use this after a system crash, or a **preserve** subcommand.

**[addr1[,addr2]]substitute/pattern/repl/ (s)****[addr1[,addr2]]substitute/pattern/repl/g**

Replaces on each specified line the *first* instance of *pattern* with the replacement pattern *repl*. If you add the **g** flag, it replaces *all* instances of *pattern* on each specified line.

**undo (u)**

Reverses the changes made in the buffer by the last buffer editing subcommand. Note that **global** subcommands are considered a single subcommand to an **undo**. You cannot **undo** a **write** or an **edit**.

**[addr1[,addr2]]write file (w)**

Writes the contents of the specified line or lines to *file*. The default range is all lines in the buffer. **edit** displays the number of lines and characters that it writes. If you do not specify a *file*, **edit** uses the current file name. If *file* does not exist, **edit** creates it.

**[addr1[,addr2]]yank [buffer] (ya)**

Places the specified line or lines in the named *buffer* (a buffer name is a single letter from a to z).

**[addr]z**

displays a screen of text, beginning with the specified line.

**[addr]z-**

Displays a screen of text, with the specified line at the bottom of the screen.

**[addr]z.** Displays a screen of text, with the specified line in the middle of the screen.

## **Related Information**

The following commands: “**ed**” on page 280, “**ex**” on page 312, and “**vi, vedit, view**” on page 832.



## env

---

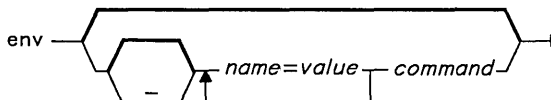
## env

---

### Purpose

Sets the environment for execution of a command.

### Syntax



OL805117

### Description

The **env** command lets you get and change your current environment, and then run the named *command* with the changed environment. Changes in the form *name = value* are added to the current environment before the command is run. If - (minus) is used, the current environment is ignored and the command runs with only the changed environment. Changes are only in effect while the named *command* is running.

If a *command* is not specified, **env** displays your current environment one *name = value* pair per line.

### Examples

1. To add a shell variable to the environment for the duration of one command:

```
TZ=MST7MDT date
env TZ=MST7MDT date
```

Each of these commands displays the current date and time in Mountain Standard Time. The two commands shown are equivalent. When **date** is finished, the previous value of **TZ** takes effect again.

2. To replace the environment with another one:

```
env - PATH=$PATH IDIR=/u/jim/include LIBDIR=/u/jim/lib make
```

This runs **make** in an environment that consists *only* of these definitions for **PATH**, **IDIR**, and **LIBDIR**. You must redefine **PATH** so that the shell can find the **make** command.

When **make** is finished, the previous environment takes effect again.

## Related Information

The following command: “**sh**” on page 637.

The **exec** system call, the **profile** file, and the **environ** miscellaneous facility in *AIX Operating System Technical Reference*.

## eqn

---

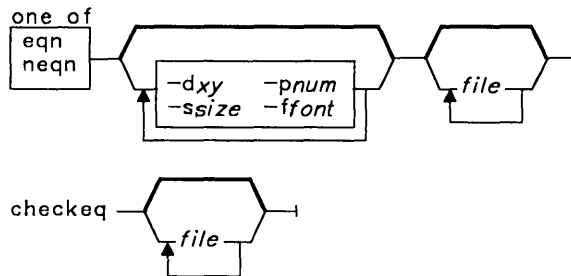
### eqn, neqn, checkeq

---

#### Purpose

Formats mathematical text for the **nroff** and **troff** commands.

#### Syntax



OL805183

#### Description

The **eqn** command is a **troff** preprocessor for typesetting mathematical text on a phototypesetter. The **neqn** command is used with **nroff** for other *printing devices*. The output of **eqn** and **neqn** is generally piped into **troff** and **nroff** as follows:

```
eqn file | troff  
neqn file | nroff
```

If you do not specify any files or if you specify - as the last file name, the commands read standard input. A line consisting of **.EQ** marks the start of equation text; the end of equation text is marked by a line consisting of **.EN**. Neither of these lines is altered by the commands, so they can be defined in macro packages to give you centering and numbering.

The program **checkeq** reports missing or unbalanced delimiter pairs and **.EQ/.EN** pairs.

For information on how to format **eqn** text, see *Text Formatting Guide*.

The **eqn** command recognizes the following mathematical words, and prints the associated symbol:

<b>above</b>	<b>dotdot</b>	<b>italic</b>	<b>rcol</b>	<b>to</b>
<b>back</b>	<b>down</b>	<b>lcol</b>	<b>right</b>	<b>under</b>
<b>bar</b>	<b>dyad</b>	<b>left</b>	<b>roman</b>	<b>up</b>
<b>bold</b>	<b>fat</b>	<b>lineup</b>	<b>rpile</b>	<b>vec</b>
<b>ccol</b>	<b>font</b>	<b>lpile</b>	<b>rpile</b>	~
<b>col</b>	<b>from</b>	<b>mark</b>	<b>size</b>	^
<b>cpile</b>	<b>fwd</b>	<b>matrix</b>	<b>sub</b>	{
<b>define</b>	<b>gfont</b>	<b>ndefine</b>	<b>sup</b>	" . . . "
<b>delim</b>	<b>gsize</b>	<b>over</b>	<b>tdefine</b>	
<b>dot</b>	<b>hat</b>	<b>pile</b>	<b>tilde</b>	

## Flags

- dxy** Sets  $x$  and  $y$  as one character delimiters of the text to be processed by **eqn**, in addition to the **.EQ** and **.EN** macros. The text between these delimiters will be treated as input to **eqn**.
- Note:** Within a file, you can also set delimiters for **eqn** text using the command **delim xy**. They are turned off by the command **delim off**. All text that is not between delimiters or **.EQ** and **.EN** is passed through unprocessed.
- ffont** Acts the same as **-s** for fonts. See the discussion of **gfont** and **font** in *Text Formatting Guide* for information on changing font within the text.
- pnum** Reduces subscripts and superscripts  $num$  points in size (the default is 3).
- ssize** Changes point size in all **eqn** processed text to  $size$ . See the discussion of **gsize** and **size** in *Text Formatting Guide* for information on changing the point size within the text.

## Related Information

The following commands: “**cw, checkcw**” on page 213, “**mm, checkmm**” on page 492, “**mmt, checkmm**” on page 495, “**nroff**” on page 525, and “**troff**” on page 526.

The **eqnchar** and **mv** miscellaneous facilities in *AIX Operating System Technical Reference*.

The discussion of **eqn** in *Text Formatting Guide*.

# errdead

---

## errdead

---

### Purpose

Extracts error records from dump.

### Syntax

```
errdead — dumpfile — { /unix | kernel-image }
```

OL805120

### Description

When the system detects a hardware error, it produces an error record containing information pertinent to the error. If **errdemon**, the error-logging demon, is not running or if the system crashes before it can place the record in the error file, the system holds the error information in a local buffer. **errdead** examines a system dump (or memory), extracts the error records, and passes them to **errpt** to generate a report. Note that no analysis is available because these error entries were never sent back via the **errdemon**.

The *dumpfile* parameter specifies the file (or memory) to be examined. The *kernel-image* parameter specifies the system name list, by default **/unix**.

### Files

<i>/unix</i>	System kernel image.
<i>/usr/bin/errpt</i>	Analysis program.
<i>/usr/tmp/err*</i>	Temporary file.

### Related Information

The following command: “**errpt**, **errpd**” on page 305.

The discussion of **errdead** in *AIX Operating System Programming Tools and Interfaces*.

# errdemon

---

## Purpose

Starts the error-logging demon.

## Syntax

`/usr/lib/errdemon` —<sup>1</sup>

---

<sup>1</sup> This command is not usually run from the command line.

OL805118

## Description

The error-logging demon **errdemon** collects error records from the operating system by reading the special file `/dev/error` and places them in one of two error log files. **errdemon** creates the names of the two log files by adding a `.0` and `.1` to the end of the file name found in `/etc/rasconf`. If an error log file does not already exist, **errdemon** creates one.

The **errdemon** command adds error records to the first error log file until it reaches the maximum allowable length specified in `/etc/rasconf`. At that point, **errdemon** closes the first error log file, changes the file name from `filename.0` to `filename.1`, and opens a new `filename.0`. Thus, the newest error records are always in `filename.0`. When it is full, **errdemon** overwrites the first file.

You can stop the error-logging demon by sending it a **SIGKILL** signal (see “**errstop**” on page 309). Normally, the `/etc/rc` command file runs **errdemon** at system start up. Only a user operating with superuser authority can start **errdemon**, and only one demon may be active at any time.

If **errdemon** is unable to log an error, it logs it in abbreviated form in `/dev/nvram`. Just one error can be logged in `/dev/nvram`, so each subsequent error overwrites any previous entries. When the system is started, **errdemon** searches for a previously written entry in `/dev/nvram` and, if a record is found, records it in one of the error log files and clears `/dev/nvram`.

## errdemon

---

### Files

/dev/error	Source of error records.
/dev/nvram	Non-volatile read-only memory.
/etc/rasconf	Configuration file.
/etc/rc	System startup file.
/usr/adm/ras/errfile*	Repository for error records.

### Related Information

The following commands: “**errpt**, **errpd**” on page 305, “**errstop**” on page 309, and “**kill**” on page 422.

The **error** and **nvram** files in *AIX Operating System Technical Reference*.

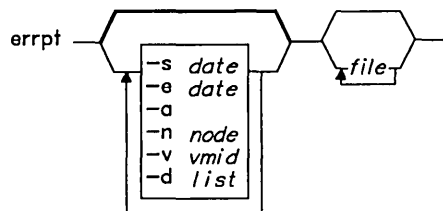
*AIX Operating System Programming Tools and Interfaces*.

## errpt, errpd

### Purpose

Processes a report of logged errors.

### Syntax



OL805410

### Description

The **errpt** command reads a specified error *file* or *files*, processes the data, and writes a report of that data to standard output. These error files should be named *file.0* or *file.1*, but do not include the *.0* or *.1* extension when you specify the file name argument. **errpt** adds the extension. If you do not specify a file name, **errpt** uses the file listed in */etc/rasconf*, adding the *.0* and *.1* extensions (these are usually */usr/adm/ras/errfile.0* and */usr/adm/ras/errfile.1*). The default report is a summary of all errors posted in the named file, as well as system information events, such as time changes, system starts, and so on.

The **errpt** command pipes error entries through the program */usr/lib/errpd*, which adds probable cause information to certain entries. If no probable cause information is added, **errpt** logs records exactly as it receives them.

### Flags

- a Produces a detailed report. This contains specific error information for every event that **errpt** formats.
- d *list* Limits the report to certain types of error records as defined by *list*. The list items can either be separated by commas or enclosed in double quotation marks and separated by commas or blanks. See “Error Identifiers” on page 306 for the valid list values.



## errpt

---

- e *date* Includes all records posted earlier than *date*, where *date* has the form *MMddhhmmyy* (month, day, hour, minute and year).
- n *nodename* Includes only entries in the error report from the specified *nodename*.
- s *date* Includes all records posted later than *date*, where *date* has the form *MMddhhmmyy*.
- v *vmid* Includes only entries in the error report from the system name specified with *vmid*.

## Error Identifiers

In the following error identifiers, **0** acts as a wildcard character, such that, for example, **H00** gives you all hardware errors (**H11** to **HFF**), and **H10** gives you all errors from **H11** to **H1F**, and so on.

### 1. Class

- H00 = Hardware (01)
- S00 = Software (02)
- I00 = IPL/Shutdown (03)
- G00 = General System Condition (04)
- U00 = User Defined, Non-Hardware

### 2. Class/Subclass

- H10 = Hardware/Processor and Memory Management Card Machine Check
- H11 = Hardware/Main Processor
- H12 = Hardware/Main Memory
- H20 = Hardware/Fixed Disk Drive and Adapter
- H30 = Hardware/Diskette Drive and Adapter
- H40 = Hardware/Tape and Adapter
- H50 = Hardware/Display Station
- H51 = Hardware/5080 Display Adapter
- H52 = Hardware/APA16 Display Adapter
- H60 = Hardware/Display Station Adapter
- H70 = Hardware/Keyboard/Mouse
- H80 = Hardware/Communication Adapters
- H81 = Hardware/RS232 Multi-port
- H84 = Hardware/Serial or Serial/Parallel
- H85 = Hardware/IBM PC Network Adapter
- H86 = Hardware/RS422 Multi-port
- H87 = Hardware/Native Serial I/O
- H8E = Hardware/SSLA
- H90 = Hardware/Parallel Printer and Adapter
- H91 = Hardware/Parallel or Serial/Parallel
- H92 = Hardware/Parallel or PC Monochrome
- HA0 = Hardware/Printers

HF0

.

.

HFF = User Defined Hardware

S10 = Software/Processor and Memory Management Card Program Check

S20 = Software/Abend

S21 = Software/Abend dump taken

S22 = Software/Abend No dump taken

S30 = Software/Program Error AIX

S33 = Software/Program Error AIX Kernel

S40 = Software/Program Error AIX Device Driver

S42 = Software/5080 Display Device Driver

S50 = Software/Program Error AIX Device Driver

S60 = Software/Program Error VRM Base

S61 = Software/Program Error VRM Attach Device

S70 = Software/Program Base VRM Component

S72 = Software/Program Base VRM Component - Virtual Terminal

S74 = Software/5080 Display VRM Device Driver

S75 = Software/5080 Peripherals VRM Device Driver Manager

S80 = Software/Program Error Application

S80 = Software/Program Error Application - Error Log Analysis

S80 = Software/Program Error Application - Interactive Workstation

S90 = Software/Program Error Application

SA0 = Software/Program Error Application

SB0 = Software/Program Error Application

SC0 = Software/Program Error Application

SD0 = Software/Program Error Application

SE0 = Software/Program Error Application

SF0 = Software/Program Error Application

I10 = IPL/Shutdown/Manual IPL

I20 = IPL/Shutdown/Soft IPL

I30 = IPL/Shutdown/Auto IPL

I40 = IPL/Shutdown/Shutdown

I50 = IPL/Shutdown/Maintenance Shutdown

G10 = General System Condition/Degraded Config

G20 = General System Condition/Set Date/ Time

G40 = General System Condition/Error Reporting

G50 = General System Condition/LPOST

G41 = General System Condition/Cause Codes

G42 = General System Condition/Device Information

G43 = General System Condition/Counters

G51 = General System Condition/Memory Test LPOST

## errpt

---

U10

.

UFF = User Defined, Non-Hardware

### errpd

The error log analysis program, `/usr/lib/errpd`, analyzes the error log data. `/usr/lib/errpd` processes error data to determine if the error is a hardware error and if the error is a temporary or permanent error.

The analysis does the following:

- Generates a number that corresponds to a service request number.
- Analyzes the data and generates the ALERT number.
- Makes the description message ID number. The description consists of the following:
  - Error Analysis determines, from the error data passed, the nature of the operation at the time of the failure. This becomes part of the error description.
  - Error Analysis determines what failed and what the error indication is. This becomes part of the error description and is used to create the ALERT number.
  - Field Replacement Unit (FRU) Analysis determines the Service Request Code. This becomes part of the error description.

### Files

`/usr/adm/ras/errfile?`      Error file.

### Related Information

The following command: “`errdemon`” on page 303.

The `errfile` file in *AIX Operating System Technical Reference*.

*AIX Operating System Programming Tools and Interfaces*.

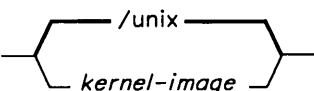
## **errstop**

---

### **Purpose**

Terminates the error-logging demon.

### **Syntax**

errstop 

OL805121

### **Description**

The **errstop** command stops the error-logging demon **errdemon** by running the **ps** command to determine the demon process ID and then sending it a Software Terminate signal (see the **signal** system call in *AIX Operating System Technical Reference*). If you do not specify *kernel-image*, **errstop** uses **/unix**. Only a user operating with superuser authority can run **errstop**.

### **Files**

**/unix**      System kernel image.

### **Related Information**

The following commands: “**errdemon**” on page 303 and “**ps**” on page 579.

The **kill** system call in *AIX Operating System Technical Reference*.

*AIX Operating System Programming Tools and Interfaces*.

## errupdate

---

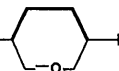
## errupdate

---

### Purpose

Updates an error report template.

### Syntax

errupdate — *file* 

OL805332

### Description

The **errupdate** command adds, replaces, or deletes error report format templates in the file */etc/errfmt*. **errupdate** creates an undo file in the current directory that it names *file.undo.err*. You can use this undo file as input to **errupdate** with the **-o** (override) flag to undo the changes **errupdate** has just made.

The **errupdate** command adds the extension **.err** to the *file* name you specify and reads update commands from the file with that name and extension. The first field of each template contains an operator:

- + To add or replace a template
- To delete a template.

If the operation is **+**, then the following fields contain the template to be replaced. If the operation is a **-**, then the second field contains the class/subclass/mask identifier of the template to delete. **errupdate** checks for valid combinations of identifiers and writes error messages if it encounters invalid combinations. When adding or replacing, it compares the version numbers of each input template with the version number of the existing template of the same class/subclass/mask and, if the version number of the input template is later, replaces the old template with the input template. If the template does not already exist, then it is added to the file. The input template *must* contain an identifier line on the first line:

\* */etc/errfmt*

or **errupdate** rejects the input file. All delete operations are performed before the add/replace operations.

## Flag

- o Does no version number checking.

## Example

The following is an example input file:

```
* /etc/errfmt
+ H87 2.0 Native Serial: IODN D2: IOCN D2: Base_Addr D4:\
    Dev_Name A4: \n: Dev_Type X4: DDI_Length D4: Error _Type X1:\
    Last_I/O X1: Line_Status X1: Printer_Status X1:
- H92
```

## Files

```
/etc/errfmt
file.err
file.undo.err
```

## Related Information

The following command: “**errpt**, **errpd**” on page 305.

*AIX Operating System Programming Tools and Interfaces.*

**ex**

---

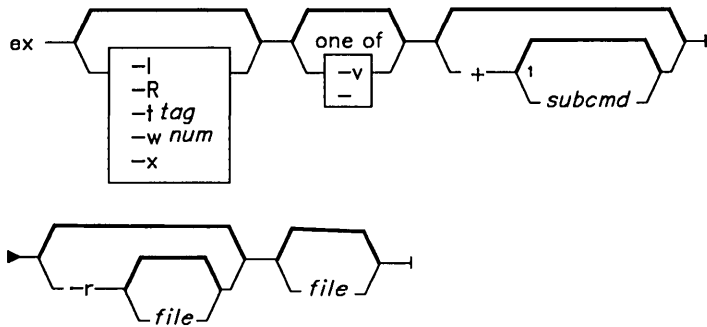
**ex**

---

## Purpose

Edits lines interactively, with screen display.

## Syntax



<sup>1</sup> Do not put a blank between these items.

OL805325

OL805308

## Description

**Warning:** The `ex` command does not support international characters. If you use this command to edit a file that contains extended characters, you can lose data.

The `ex` command is a line-oriented text editor that is a subset of the `vi` screen editor. The `ex` command is similar to `ed`, but is more powerful, providing multi-line displays and access to a screen editing mode. You may prefer to call `vi` directly to have environmental variables set for screen editing.

**Note:** Some `vi` subcommands have meanings that differ from `ed` subcommands.

A limited subset of `ex`, `edit`, is available for novice or casual use (see “`edit`” on page 292).

To determine how to drive your work station more efficiently, `ex` uses the work station capability data base `terminfo` and the type of the work station you are using from the shell environment variable `TERM`.

Some features of **ex** are:

- You can view text in files. The **z** subcommand lets you access windows of text, and you can scroll through text by pressing **Ctrl-D** and **Ctrl-U**. The **vi** subcommand provides further viewing options and active screen-editing by invoking the **vi** editor.
- The **undo** subcommand lets you revoke the last previous subcommand entered (except for **q** and **w**). **undo** can be used to revoke itself. You can switch back and forth between the latest change in the edit file and the last prior file status; you can view the effect of a subcommand without having irrevocably performed it. (**ex** displays changed lines and indicates when more than a few lines are affected by a subcommand.)

**Note:** The **undo** subcommand causes all marks to be lost on lines changed and then restored if the marked lines were changed. It never clears the buffer modified condition.

- If the system or the editor crashes, you can retrieve your work (except changes that were in the buffer) by re-entering the editor using the **-r** parameter and providing the file name. When the file name is not specified, all open files in your partition are listed.
- You can queue a sequence or group of files to edit. List the files on the **ex** command line and use the **next** subcommand to access each file sequentially. Within **ex**, you can give **next** subcommand a list of file names or a pattern (as used by the shell) to specify a new set of files to deal with. In general, you can designate file names to the editor using all of the pattern-matching symbols that the shell will accept. The wild card character **%** is available for forming file names and represents the name of the current edit file.
- A group of buffers, named **a** through **z**, lets you move text between files and within a file. You can temporarily place text in named buffers and copy or reinsert it in a file or carry it over to another file you edit. The buffers are cleared when you finally quit the editor.

**Warning:** **ex** does not notify you if text is placed in a buffer and not used before exiting the editor.

- The **ex** utility lets you use patterns that match words. This lets you, for example, search only for the word “ink” when your document also contains the word “inkblot” or “blink.”
- The **z** subcommand displays a window of logical lines. You can select the number of lines displayed and locate the current line within the display simultaneously.

**Note:** More than a screen of output can result when the file lines are longer than the output display lines because the set number of logical lines are displayed rather than a number of physical lines.



## ex States

- Command** Normal and initial state. Input is prompted for by : (colon). Pressing END OF FILE (Ctrl-D) clears an uncompleted subcommand from the command line.
- Entry** Entered by **a**, **i** and **c**. In this state you can enter text. Entry state ends normally with a line that has only a . (period) on it or ends abnormally if you press INTERRUPT (Alt-Pause).
- Visual** Entered by **v** or **o**, and returns to command state with **Q** or **^\**.

## ex Command Names and Abbreviations

**Note:** Most of the following commands are discussed under “**edit**” on page 292 or “**vi**, **vedit**, **view**” on page 832.

abbrev	<b>ab</b>	next	<b>n</b>	unabbrev	<b>una</b>
append	<b>a</b>	number	<b>nu</b>	undo	<b>u</b>
args	<b>ar</b>			unmap	<b>unm</b>
change	<b>c</b>	preserve	<b>pre</b>	version	<b>vc</b>
copy	<b>co</b>	print	<b>p</b>	visual	<b>vi</b>
delete	<b>d</b>	put	<b>pu</b>	write	<b>w</b>
edit	<b>e</b>	quit	<b>q</b>	xit	<b>x</b>
file	<b>f</b>	read	<b>re</b>	yank	<b>ya</b>
global	<b>g</b>	recover	<b>rec</b>	window	<b>z</b>
insert	<b>i</b>	rewind	<b>rew</b>	escape	<b>!</b>
join	<b>j</b>	set	<b>se</b>	lshift	<b>&lt;</b>
list	<b>l</b>	shell	<b>sh</b>	print next	<b>CR</b>
map		source	<b>so</b>	resubst	<b>&amp;</b>
mark	<b>ma</b>	stop	<b>st</b>	rshift	<b>&gt;</b>
move	<b>m</b>	substitute	<b>s</b>	scroll	<b>^D</b>

## Subcommand Addresses

<b>\$</b>	The last line	<i>x-num</i>	The <i>num</i> th line before <i>x</i>
<b>+</b>	The next line	<i>x,y</i>	Lines <i>x</i> through <i>y</i>
<b>-</b>	The previous line	<i>'m</i>	The line marked with <i>m</i>
<b>+ num</b>	The <i>num</i> th line forward	<i>"</i>	The previous context
<b>- num</b>	The <i>num</i> th previous line	<i>/\$pat</i>	The next line with <i>pat</i> at end of line

---

%	The first through last lines	/^ <i>pat</i>	The next line with <i>pat</i> at start of line
<i>num</i>	line <i>num</i>	/ <i>pat</i>	The next line with <i>pat</i>
.	The current line	? <i>pat</i>	The previous line with <i>pat</i>

### Scanning Pattern Formation

^	The beginning of the line
\$	The end of the line
.	Any character
\<	The beginning of the word
\>	The end of the word
[ <i>string</i> ]	Any character in <i>string</i>
[^ <i>string</i> ]	Any character not in <i>string</i>
[ <i>x-y</i> ]	Any character between <i>x</i> and <i>y</i> , inclusive
*	Any number of the preceding character.

### Flags

- l Indents appropriately for Lisp code, and accepts the () {} [[ and ]] characters as text rather than interpreting them as **vi** subcommands. The *Lisp* modifier is active in **open** or **visual** modes.
- r [*file*] Recovers *file* after an editor or system crash. If you do not specify *file*, a list of all saved files is displayed.
- R The **readonly** option is set, preventing you from altering the file.
- t *tag* Loads the file that contains *tag* and positions the editor at *tag*.
- v Invokes the **visual** editor.  
**Note:** When the **v** flag is selected, an enlarged set of subcommands are available, including screen editing and cursor movement features. See “**vi**, **vedit**, **view**” on page 832.
- Suppresses all interactive-user feedback. If you use this flag, file input/output errors do not generate a helpful error message.
- +*subcmd* Begins the edit at the specified editor search or subcommand. When *subcommand* is not entered, + places the current line to the bottom of the file. Normally **ex** sets current line to the start of the file, or to some specified tag or pattern.

## Files

<code>/usr/lib/ex?.?strings</code>	Error messages.
<code>/usr/lib/ex?.?recover</code>	Recover subcommand.
<code>/usr/lib/ex?.?preserve</code>	Preserve subcommand.
<code>/usr/lib/*/*</code>	Describes capabilities of work stations.
<code>\$HOME/.exrc</code>	Editor startup file.
<code>./exrc</code>	Editor startup file.
<code>/tmp/Exnnnnn</code>	Editor temporary.
<code>/tmp/Rxnnnnn</code>	Names buffer temporary.
<code>/usr/preserve</code>	Preservation directory.

## Related Information

The following commands: “**vi**, **vedit**, **view**” on page 832, “**edit**” on page 292, “**awk**” on page 70, “**ed**” on page 280, “**grep**” on page 381, and “**sed**” on page 629.

The **curses** subroutine and the **TERM**, **INIT**, and **terminfo** files in *AIX Operating System Technical Reference*.

---

**expr**


---

**Purpose**

Evaluates arguments as an expression.

**Syntax**

*expr* — *expression* —|

OL805048

**Description**

The **expr** command reads an *expression*, evaluates it, and writes the result to standard output. Within *expression*, you must separate each term with blanks, precede characters special to the shell with a backslash (\), and quote strings containing blanks or other special characters. Note that **expr** returns 0 to indicate a zero value, rather than the null string. Integers may be preceded by a unary minus sign. Internally, integers are treated as 32-bit, two's complement numbers.

The operators and keywords are described in the following listing. Characters that need to be escaped are preceded by a backslash (\). The list is in order of increasing precedence, with equal precedence operators grouped within braces ({}).

*expression1* \| *expression2*

Returns *expression1* if it is neither null nor 0; otherwise it returns *expression2*.

*expression1* \& *expression2*

Returns *expression1* if neither *expression1* nor *expression2* is null or 0; otherwise it returns 0.

*expression1* { =, |>, |>=, |<, |<=, != } *expression2*

Returns the result of an integer comparison if both expressions are integers; otherwise returns the result of a string comparison.

*expression1* {+, - } *expression2*

Adds or subtracts integer-valued arguments.

*expression1* { \\*, /, % } *expression2*

Multiplies, divides, or provides the remainder from the division of integer-valued arguments.

## expr

---

*expression1* : *expression2*

Compares *expression1* with *expression2*, which must be a pattern; pattern syntax is the same as that of the **ed** command (see page 280), except that all patterns are **anchored**, so **^** (which anchors a pattern to the beginning of a line), is not a special character in this context.

Normally, the matching operator returns the number of characters matched. Alternatively, you can use the **\( . . . \)** symbols in *expression2* to return a portion of *expression1*. In an expression such as **[a-z]**, the minus means “through” according to the current collating sequence. A collating sequence may define **equivalence classes** for use in character ranges. See the “Overview of International Character Support” in *Managing the AIX Operating System* for more information on collating sequences and equivalence classes.

The **expr** command returns the following exit values:

- 0 The expression is neither null nor 0.
- 1 The expression is null or 0.
- 2 The expression is invalid.

**Note:** After parameter processing by the shell, **expr** cannot distinguish between an operator and an operand except by the value. Thus, if **\$a** is **=**, the command:

```
expr $a = '='
```

looks like:

```
expr = = =
```

after the shell passes the arguments to **expr**, and they will all be taken as the **=** operator. The following works:

```
expr X$a = X=
```

## Examples

1. To modify a shell variable:

```
COUNT=`expr $COUNT + 1`
```

This adds 1 to the shell variable **COUNT**. The **expr** command is enclosed in grave accents, which causes the shell to substitute the standard output from **expr** into the **COUNT=** command. For more details, see “Command Substitution” on page 647.

2. To find the length of a shell variable:

```
LENGTH=`expr $STR : ".*"`
```

This sets LENGTH to the value given by the : (colon) operator. The pattern .\* matches any string from beginning to end, so the colon operator gives the length of STR as the number of characters matched. Note that ".\*" must be in quotes to prevent the shell from treating the \* as a pattern-matching character. The quotes themselves are not part of the pattern.

If STR is set to the null string, the error message `expr: syntax error` is displayed. This happens because the shell does not normally pass null strings to commands. In other words, the **expr** command sees only

```
: .*
```

(The shell also removes the quotation marks.) This does not work because the colon operator requires two values. We can fix this problem by enclosing the shell variable in double quotation marks:

```
LENGTH=`expr "$STR" : ".*"``
```

Now if STR is null, LENGTH is set to zero. Enclosing shell variables in double quotes is recommended in general. However, do *not* enclose shell variables in single quotes. See page 641 for details about quoting.

3. To use part of a string:

```
FLAG=`expr "$FLAG" : "-*\(.*\)"``
```

This removes leading minus signs, if any, from the shell variable FLAG. The colon operator gives the part of FLAG matched by the part of the pattern enclosed in `\( \)`. If you omit the `\( \)`, the colon operator gives the number of characters matched.

If FLAG is set to - (minus), a syntax error message is displayed. This happens because the shell substitutes the value of FLAG before running the **expr** command. **expr** does not know that the minus is the value of a variable. It can only see:

```
- : -*\(.*\)
```

and it interprets the first minus sign as the subtraction operator. We can fix this problem by using:

```
FLAG=`expr "x$FLAG" : "x-*\(.*)"``
```

## expr

---

4. To use **expr** in an **if** statement:

```
if expr "$ANSWER" : "[yY]" >/dev/null
then
    # ANSWER begins with "y" or "Y"
fi
```

If ANSWER begins with y or Y, the **then** part of the **if** statement is performed. If the match succeeds, the result of the expression is 1 and **expr** returns an exit value of 0, which is recognized as the logical value TRUE by **if**. If the match fails, the result is 0 and the exit value 1 (FALSE).

Redirecting the standard output of **expr** to the **/dev/null** special file discards the result of the expression. If you do not redirect it, the result is written to the standard output, which is usually your work station display.

5. Consider the following expression:

```
expr "$STR" = "="
```

If STR has the value = (equal sign), then after the shell processes this command **expr** sees the expression:

```
= = =
```

The **expr** command interprets this as three = operators in a row and displays a syntax error message. This happens whenever the value of a shell variable is the same as one of the **expr** operators. You can avoid this problem by doing the following:

```
expr "x$STR" = "x="
```

## Related Information

The following commands: “**ed**” on page 280 and “**sh**” on page 637.

The “Overview of International Character Support” in *Managing the AIX Operating System*.

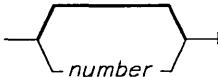
## factor

---

### Purpose

Factors a number.

### Syntax

factor 

OL805051

### Description

When called without an argument, the **factor** command waits for you to enter a positive number less than  $2^{56}$ . It then writes the prime factors of that number to standard output. It displays each factor the proper number of times. To exit, enter a 0 or any nonnumeric character.

When called with an argument, **factor** determines the prime factors of *number*, writes the results to standard output, and exits.

### Example

To calculate the prime factors of 123:

```
factor 123
```

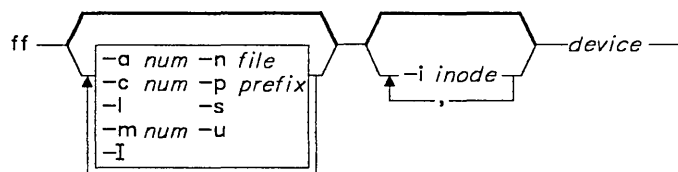
This displays:

```
123
  3
 41
```



**ff****ff****Purpose**

Lists the file names and statistics for a file system.

**Syntax**

OL805122

**Description**

**Warning:** This program is not intended for use with diskette-based file systems because of the difference in superblock structure and the general format of the file system.

The **ff** command reads the i-list and directories specified by *device* and writes information about them to standard output. It assumes that *device* is a file system, and saves i-node data for files specified by flags. The output from the **ff** command consists of the path name for each saved i-node, in addition to other file information that you request with the flags. The output is listed in order by i-node number, with tabs between all fields. The default line produced by **ff** includes the path name and i-number fields. With all flags enabled, the output fields include path name, i-number, size, and UID.

The *num* parameter in the flags descriptions is a decimal number, where *+num* means more than *num*, *-num* means less than *num*, and *num* means exactly *num*. A day is defined as a 24-hour period.

The **ff** command lists only a single path name out of many possible ones for an i-node with more than one link, unless you specify the **-l** flag. With **-l**, **ff** applies no selection criteria to the names listed. All possible names for every linked file on the file system are included in the output. On very large file systems, memory may run out before **ff** does.

---

## Flags

- a** *num*     Selects if the i-node has been accessed in *num* days.
- c** *num*     Selects if the i-node has been changed in *num* days.
- i** *i-node*    Generates names for only those i-nodes specified in the *inode* list.
- I**            Does not display the i-node number after each path name.
- l**            Generates a list of all path names for files with more than one link.
- m** *num*     Selects if the file associated with the i-node has been modified in *num* days.
- n** *file*     Selects if the file associated with the i-node has been modified more recently than the specified *file*.
- p** *prefix*   Adds the specified *prefix* to each path name. The default prefix is . (dot).
- s**            Writes the file size, in bytes, after each path name.
- u**            Writes the owner's login name after each path name.

## Examples

1. To list the path names of all files in a given file system:

```
ff -I /dev/hd0
```

This displays the path names of the files on the /dev/hd0 disk. If you do not specify the -I flag, then **ff** also displays the i-number of each file.

2. To list files that have been modified recently:

```
ff -m -2 -u /dev/hd0
```

This displays the path name, i-number, and owner's user name (-u) of each file on /dev/hd0 that has been modified within the last two days (-m -2).

3. To list files that have **not** been used recently:

```
ff -a +30 /dev/hd0
```

This displays the path name and i-number of each file that was last accessed more than 30 days ago (-a +30).

4. To find out the path names of certain i-nodes:

```
ff -l -i 451,76 /dev/hd0
```

This displays all the path names (-l) associated with i-nodes 451 and 76.

## **Related Information**

The following commands: “**find**” on page 326 and “**ncheck**” on page 505.



# file

---

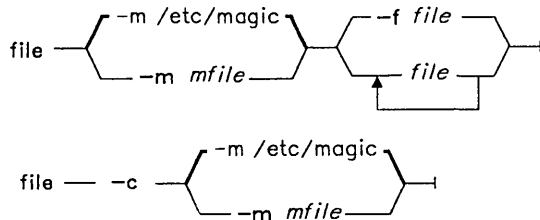
# file

---

## Purpose

Determines file type.

## Syntax



OL805124

## Description

The **file** command reads its input *files*, performs a series of tests on each one, and attempts to classify them by their types. **file** then writes the file types to standard output. If a file appears to be ASCII, **file** examines the first 512 bytes and tries to determine its language. If a file does not appear to be ASCII, **file** further attempts to distinguish a binary data file from a text file that contains extended characters. If *file* is an **a.out** file, and the version number is greater than zero (see “**ld**” on page 427), **file** displays the version stamp.

The **file** command uses the file **/etc/magic** to identify files that have some sort of **magic number**, that is, any file containing a numeric or string constant that indicates its type. Comments at the beginning of **/etc/magic** explain its format.

## Flags

- c** Checks the *mfile* (**/etc/magic** by default) for format errors. This validation is not normally done. File typing is not done under this flag.
- f file** Reads *file* for a list of files to examine.
- m mfile** Specifies *mfile* as the magic file (**/etc/magic** by default).

## Examples

1. To display the type of information a file contains:

```
file myfile
```

This displays the file type of `myfile` (directory, data, ASCII text, C-program source, archive, and so forth).

2. To display the type of each file named in a list of file names:

```
file -f filenames
```

This displays the type of each file with a name that appears in `filenames`. Each file name must appear alone on a line.

To create `filenames`:

```
ls >filenames
```

then edit `filenames` as desired.

## Files

`/etc/magic` File type database.

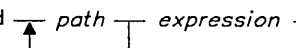
## Related Information

“Overview of International Character Support” in *Managing the AIX Operating System*.

**find****find****Purpose**

Finds files matching expression.

**Syntax**

```
find  path expression →
```

OL805125

**Description**

The **find** command recursively searches the directory tree for each specified *path*, seeking files that match a Boolean *expression* written using the terms given below. The output from **find** depends on the terms used in *expression*.

**Expression Terms**

In the following descriptions, the parameter *num* is a decimal integer that can be specified as *+num* (more than *num*), *-num* (less than *num*), or *num* (exactly *num*).

**-name** *file* True if *file* matches the file name. You can use pattern-matching characters, provided they are quoted. In an expression such as [**a-z**], the minus means “through” according to the current collating sequence. A collating sequence may define **equivalence classes** for use in character ranges. See “Overview of International Character Support” in *Managing the AIX Operating System* for more information on collating sequences and equivalence classes.

**-node** *nname* True if the file resides in the node *nname*. If *nname* is a valid nickname, it is used as is. If *nname* is not a valid nickname but has a valid NID syntax, it is used as a NID.

**-perm** *onum* True if the file permission code of the file exactly matches the octal number *onum* (see “**chmod**” on page 128 for an explanation of file permissions). The *onum* parameter may be up to three octal digits.

If you want to test the higher-order permission bits (the set-user-ID bit or set-group-ID bit, for example), prefix the *onum* parameter with a minus (-) sign. This makes more flag bits significant (see the **stat** system call for an explanation of the additional bits), and also changes the comparison to:

- 
- (*flags&onum*)= *onum*
- type** *type* True if the file *type* is of the specified type as follows:
- b** Block special file
  - c** Character special file
  - d** Directory
  - f** Plain file
  - p** FIFO (a named pipe).
- links** *num* True if the file has *num* links. See “**ln**” on page 450.
- user** *uname* True if the file belongs to the user *uname*. If *uname* is numeric and does not appear as a login name in the **/etc/passwd** file, it is interpreted as a user ID.
- group** *gname* True if the file belongs to the group *gname*. If *gname* is numeric and does not appear in the **/etc/group** file, it is interpreted as a group ID.
- size** *num* True if the file is *num* blocks long (512 bytes per block). For this comparison the file size is rounded up to the nearest block.
- atime** *num* True if the file has been accessed in *num* days.
- mtime** *num* True if the file has been modified in *num* days.
- ctime** *num* True if the file i-node has been changed in *num* days.
- exec** *cmd* True if the *cmd* runs and returns a zero value as exit status. The end of *cmd* must be punctuated by a quoted or escaped semicolon. A command parameter **{}** is replaced by the current path name.
- ok** *cmd* The **find** command asks you whether it should start *cmd*. If your response begins with *y*, *cmd* is started. The end of *cmd* must be punctuated by a quoted or escaped semicolon.
- print** Always true; causes the current path name to be displayed. **find** does not display path names unless you specify this expression term.
- cpio** *device* Write the current file to *device* in cpio format. See “**cpio**” on page 158.
- newer** *file* True if the current file has been modified more recently than the file indicated by *file*.
- depth** Always true. This causes the descent of the directory hierarchy to be done so that all entries in a directory are affected before the directory itself. This can be useful when **find** is used with **cpio** to transfer files that are contained in directories without write permission.
- \( *expression* \)** True if the expression in parentheses is true.



## find

---

You may perform the following logical operations on these terms (listed in order of decreasing precedence):

- Negate a term (! is the NOT operator).
- Concatenate terms (juxtaposing two terms implies the AND operation).
- Alternate terms (-o is the OR operator).

## Examples

1. To list all files in the file system with a given base file name:

```
find / -name .profile -print
```

This searches the entire file system and writes the complete path names of all files named `.profile`. The `/` tells **find** to search the root directory and all of its subdirectories. This may take a while, so it is best to limit the search by specifying the directories where you think the files might be.

2. To list the files with a specific permission code in the current directory tree:

```
find . -perm 0600 -print
```

This lists the names of the files that have *only* owner-read and owner-write permission. The `.` (dot) tells **find** to search the current directory and its subdirectories. See “**chmod**” on page 128 for details about permission codes.

3. To search several directories for files with certain permission codes:

```
find manual clients proposals -perm -0600 -print
```

This lists the names of the files that have owner-read and owner-write permission *and possibly other permissions*. The directories `manual`, `clients`, and `proposals`, and their subdirectories, are searched. Note that `-perm 0600` in the previous example selects only files with permission codes that match `0600` *exactly*. In this example, `-perm -0600` selects files with permission codes that allow *at least* the accesses indicated by `0600`. This also matches the permission codes `0622` and `2744`.

4. To search for regular files with multiple links:

```
find . -type f -links +1 -print
```

This lists the names of the ordinary files (`-type f`) that have more than one link (`-links +1`). Note that every directory has at least two links: the entry in its parent directory and its own `.` (dot) entry. See “**ln**” on page 450 for details about multiple file links.

5. To back up selected files in **cpio** format:

```
find . -name "*.c" -cpio /dev/rfd0
```

This saves all the .c files onto the diskette in **cpio** format. See “**cpio**” on page 158 for details. Note that the pattern “\*.c” must be quoted to prevent the shell from treating the \* as a pattern-matching character. This is a special case in which **find** itself decodes the pattern-matching characters.

6. To perform an action on all files that meet complex requirements:

```
find . \( -name a.out -o -name "*.o" \) -atime +7 -exec rm {} \;
```

This deletes (-exec rm {} \;) all files named a.out or that end with .o, and that were last accessed over seven days ago (-atime +7). The -o flag is the logical OR operator.

## Files

```
/etc/passwd
/etc/group
```

## Related Information

The following commands: “**cpio**” on page 158, “**sh**” on page 637, and “**test**” on page 750.

The **stat** system call and the **cpio** and **fs** files in *AIX Operating System Technical Reference*.

“Overview of International Character Support” and “Using Distributed Services” in *Managing the AIX Operating System*.

# fish

---

## fish

---

### Purpose

Plays the card game Go Fish.

### Syntax

`/usr/games/fish` —

OL805189

### Description

The object of the **fish** game is to accumulate *books* of four cards with the same face value. You and the program take turns asking each other for a card in your hand. If your opponent has any of that card, he must hand them over. If not, he says GO FISH, and you draw a card from the pool of undealt cards. If you draw the card you asked for, you draw again. As books are made, they are laid down on the table. Play continues until there are no cards left. The player with the largest number of books wins the game. **fish** tells you the winner and exits.

The **fish** game asks if you want instructions before play begins. To see the instructions, enter *y* or *YES*.

Entering *p* as your first move gives you the professional level game.

The **fish** game tells you the cards in your hand each time it prompts for a move. It tells you when either side makes a book, says GO FISH for you, and draws for you. All you must enter as play progresses is the value of the card you want to ask for. If you press only the **Enter** key, you are given information about the number of cards in your opponent's hand and in the pool.

To exit the game before play is completed, press INTERRUPT (**Alt-Pause**).

---

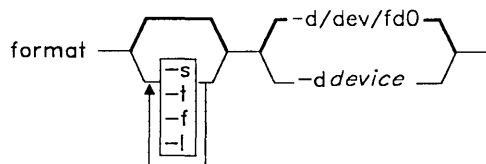
# format

---

## Purpose

Formats diskettes.

## Syntax



OL805395

## Description

The **format** command formats diskettes in the specified *device* (**/dev/fd0** by default). **format** determines the device type, either a 360K or a 1.2M diskette drive. By default, it formats a diskette in a 360K drive to have 40 cylinders, 9 sectors per track, and 2 sides and a diskette in a 1.2M drive to have 80 cylinders, 15 sectors per track, and 2 sides.

## Flags

- ddevice** Specifies the device containing the diskette to be formatted.
- f** Formats the diskette without checking for bad tracks, thus formatting the diskette faster.
- l** Formats a 360K diskette in a 1.2M diskette drive.  
**Warning:** A 360K diskette drive may not be able to read a 360K diskette that has been formatted in a 1.2M drive.
- s** Specifies a single-sided diskette. Use only for 360K diskette drives.
- t** Specifies that the number of sectors on a 360K diskette should be 8.

## Related Information

The **fd** file in *AIX Operating System Technical Reference*.

# fortune

---

## fortune

---

### Purpose

Tells a fortune.

### Syntax

`/usr/games/fortune`  $\rightarrow$

OL805190

### Description

The **fortune** game tells a fortune, selected at random from the file `/usr/games/lib/fortunes`, and exits.

You can edit the file `/usr/games/lib/fortunes` to add your own fortunes. Each saying in the file should be a single line. **fortune** folds long sayings into multiple lines as necessary.

### Files

`/usr/games/lib/fortunes`

**fptype**

---

**Purpose**

Displays the floating point configuration of the system.

**Syntax**

fptype —

OL805471

**Description**

The **fptype** command calls a subroutine which determines the current floating point configuration. **fptype** returns one of the following values and displays the corresponding message:

**Return Value Message**

0	Floating Point Type = Software Emulation
1	Floating Point Type = FPA Card
2	Floating Point Type = APC Card with MC68881
4	Floating Point Type = AFPA - No DMA Support
12	Floating Point Type = AFPA - With DMA Support

**Related Information**

The **fpfp** subroutine in *AIX Operating System Technical Reference*.

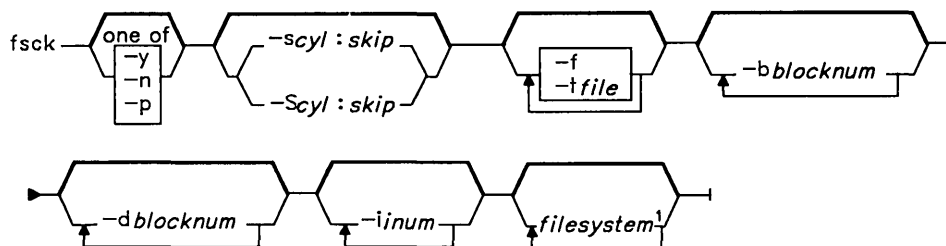


## fsck, dfscck

### Purpose

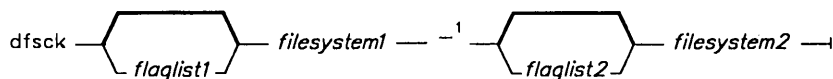
Checks file system consistency and interactively repairs the file system.

### Syntax



<sup>1</sup>The default action is to check every file system with the attribute `check=true` in the file `/etc/filesystem`.

OL8055384



<sup>1</sup>Use a `-` to separate the groups when you specify flags as part of the argument.

OL805456

### Description

**Warning:** Always run **fsck** on file systems after a system crash. Corrective actions may result in some loss of data. The default action for each consistency correction is to wait for the operator to respond yes or no. If you do not have write permission for an affected file, **fsck** defaults to a no response in spite of your actual response.

The **fsck** command checks and interactively repairs inconsistent *filesystems*. It should be run on every file system as part of system initialization (see “**rc**” on page 594). You must have superuser authority to run **fsck**. Normally, the file system is consistent, and **fsck**



## fsck

---

merely reports on the number of files, used blocks and free blocks in the file system. If the *filesystem* is inconsistent, **fsck** displays information about the inconsistencies found and prompts you for permission to repair them. **fsck** is conservative in its repair efforts and tries to avoid actions that might result in the loss of valid data. In certain cases, however, **fsck** recommends the destruction of a damaged file.

If you do not specify *filesystem*, **fsck** looks at */etc/filesystems* to find a list of file systems to check by default. **fsck** can perform checks (on separate arms) in parallel (running in parallel processes). This can reduce the time required to check a large number of file systems.

In */etc/filesystems*, automatic checking may be enabled by adding a line in the stanza, as follows:

```
check=true
```

If you specify the **-p** flag, **fsck** can perform multiple checks at the same time. To tell **fsck** which file systems are on the same drives change the **check** specification in */etc/filesystems* as follows:

```
check=number
```

The *number* tells **fsck** which group contains a particular file system. File systems on a single drive are placed in the same group. Each group is checked in a separate parallel process. File systems are checked, one at a time, in the order that they appear in */etc/filesystems*. All **check=true** file systems are in group 1. **fsck** attempts to check the root file system before any other file system regardless of order specified on the command line or in */etc/filesystems*.

The **fsck** command checks for the following inconsistencies:

- Blocks allocated to multiple files or to a file and the free list.
- Blocks allocated to a file or on the free list outside the range allowable block numbers.
- Discrepancies between the number of directory references to a file and the link count in the file.
- Size checks:
  - Incorrect number of blocks.
  - Directory size not 16-byte aligned.
- Bad i-node format.
- Blocks not accounted for anywhere.
- Directory checks:
  - File pointing to an i-node that is not allocated.
  - I-node number out of range.
  - Dot (.) link missing or not pointing to itself.
  - Dot dot (..) link missing or not pointing to the parent directory.
  - Files that are not referenced or directories that are not reachable.
- Superblock checks:
  - More than 65535 i-nodes.
  - More blocks for i-nodes than there are in the file system.

- Bad free block list format.
- Total free block and/or free i-node count incorrect.

Orphaned files and directories (those that cannot be reached) are, if you allow it, reconnected by placing them in the **lost + found** subdirectory in the root directory. The name assigned is the i-node number. The only restriction is that the directory **lost + found** must already exist in the root directory of the file system being checked and must have empty slots in which entries can be made (accomplished by copying a number of files to the directory and then removing them before you run **fsck**). If you do not allow **fsck** to reattach an orphaned file, it requests permission to destroy the file. When **fsck** displays i-node information, the **NLTIME** environment variable controls the format of the modification time.

In addition to its messages, **fsck** records the outcome of its checks and repairs through its exit value. This exit value can be any sum of the following conditions:

- 0 All checked file systems are now ok.
- 2 **fsck** was interrupted before it could complete checks or repairs.
- 4 **fsck** changed the mounted file system; the user must restart the system immediately.
- 8 The file system contains unrepaired damage.

When the system is being started up normally, **fsck** runs with the **-p** flag from **/etc/rc** (see “**rc**” on page 594). If **fsck** detects and repairs errors on the root or other mounted file systems, it displays a message on the console and restarts the system, if possible. If it cannot restart the system or if it detects errors that it cannot repair, it displays appropriate messages on the console and returns an exit value indicating that an immediate restart is necessary.

**Note:** All statistics reported by **fsck** are in 512-byte blocks, regardless of the actual block size of the file system being checked. All user specifications should be specified in 512-byte blocks.

## dfscck

The **dfscck** command lets you simultaneously check two file systems on two different drives. Use the *flaglist1* and *flaglist2* arguments to pass flags and parameters for the two sets of file systems. Use a **-** (minus) to separate the file system groups if you specify flags as part of the arguments.

The **dfscck** command permits you to interact with two **fsck** commands at once. To aid in this, **dfscck** displays the file system name with each message. When responding to a question from **dfscck**, you must prefix your response with a **1** or a **2** to indicate whether the answer refers to the first or second file system group.

**Note:** Do not use **dfscck** to check the root file system (**/dev/hd0**).

# fsck

---

## Flags

- bblocknum** Designates a block as bad. **fsck** searches for any files that contain the specified block. If it finds any such files, it asks permission to delete them. If it finds no such files or is told to delete all such files, the specified block is added to the bad block list in i-node 1. This keeps the block out of circulation so that it cannot be allocated to any user file.
- dblocknum** Searches for references to a specified disk block. Whenever **fsck** encounters a file that contains a specified block, it displays the i-node number and all path names that refer to it.
- f** Performs a fast check. Under normal circumstances, the only file systems likely to be affected by halting the system without shutting down properly are those that were mounted when the system stopped. the **-f** flag tells **fsck** not to check file systems that were cleanly unmounted. **fsck** determines this by inspecting the **s\_fm** flag in the file system superblock. this flag is set whenever a file system is mounted and cleared when it is cleanly unmounted. if a file system was cleanly unmounted, it is unlikely to have any problems. because most file systems are cleanly unmounted, not checking those file systems can reduce the checking time.
- iinum** Searches for references to a specified i-node. Whenever **fsck** encounters a directory reference to a specified i-node number, it displays the full path name of the reference.
- n** Assumes a NO response to all questions asked by **fsck**; does not open *filesystem* for writing.
- p** Does not display messages about minor problems, but fixes them automatically. This flag does not grant the wholesale license that the **-y** flag does and is useful for performing automatic checks when the system is to be started normally. You should use this flag whenever the system is being run automatically as part of the system startup procedures.
- s[cyl:skip]** Ignores the actual free list and unconditionally reconstructs a new one. You can specify an optional interleave specification with this flag: *cyl* specifies the number of blocks per cylinder; *skip* specifies the number of blocks to skip. If you do not specify *cyl* or *skip*, **fsck** uses the interleave parameters in the superblock. The file system should be unmounted while this is done; if this is not possible, be sure that you are running no programs and that you perform a system restart immediately afterwards so that the old copy of the superblock in memory is not written to disk.
- S[cyl:skip]** Conditionally reconstructs the free list. This flag is like the **-s** flag except that the free list is rebuilt only if there are no discrepancies discovered in the file system. Using **-S** forces a NO response to all questions asked by **fsck**. Use this flag to force free list reorganization on uncontaminated file systems.

- tfile** Uses *file* as a scratch file if **fsck** cannot obtain enough memory to keep its tables. If you do not specify **-t** and **fsck** needs a scratch file, it prompts you for the name of the scratch file. However, if you have specified the **-p** flag, **fsck** fails. The file chosen must not be on the file system being checked. If it is not a special file, it is removed when **fsck** ends.
- y** Assumes a **yes** response to all questions asked by **fsck**. This lets **fsck** take any action that it considers necessary. Use this flag only on severely damaged file systems.

## Examples

1. To check all the default file systems:

```
fsck
```

This checks all the file systems marked `check=true` in `/etc/filesystems`. This form of the **fsck** command asks you for permission before making any changes to a file system.

2. To fix minor problems with the default file systems automatically:

```
fsck -p
```

3. To check a specific file system:

```
fsck /dev/hd1
```

This checks the unmounted file system located on the `/dev/hd1` device.

4. To simultaneously check two file systems on two different drives:

```
dfsk -p /dev/hd1 - -p /dev/hd7
```

This checks both file systems simultaneously, if the file systems on the devices `/dev/hd1` and `/dev/hd7` are located on two different drives. You can also specify the file system names that are found in the `/etc/filesystems` file.

## Files

`/etc/filesystems` Contains default list of file systems to check.

## Related Information

The following commands: “**rc**” on page 594, “**fsdb**” on page 338, “**istat**” on page 415, “**mkfs**” on page 487, “**ncheck**” on page 505, and “**shutdown**” on page 663.

The `filesystems` and `fs` files in *AIX Operating System Technical Reference*.

The discussion of **fsck** and **dfsk** in *Managing the AIX Operating System*

# fsdb

---


## fsdb

---

### Purpose

Debugs file systems.

### Syntax

fsdb — filesystem 

OL805244

### Description

**Warning:** This program is not intended for use with diskette-based file systems because of the difference in superblock structure and the general format of the file system.

You can use the **fsdb** command to examine and patch a damaged file system after a system crash. It allows you to access blocks and i-numbers and to examine various parts of an i-node. You can reference components of the i-node symbolically. These features simplify procedures for correcting control-block entries or for descending the file-system tree.

The file system to be examined can be specified by a block device name, a raw device name, or a mounted file system name. In the latter case, **fsdb** determines the associated file name by reading the file */etc/filesystems*.

Any numbers you enter are considered decimal by default, unless you prefix them with a 0 (zero) to indicate an octal number.

Because **fsdb** reads and writes one block at a time, it works with raw as well as with block I/O. It uses a buffer management routine to retain commonly used blocks of data in order to reduce the number of **read** system calls. All assignment operations write the corresponding block immediately.

### Flag

- Disables the error checking routines used to verify i-node and block addresses. The **O** subcommand toggles these routines on and off. When these routines are running, **fsdb** reads the i-size and f-size entries from the superblock of the file system.

## Subcommands

The subcommands you give to **fsdb** are requests to display or modify information. A display subcommand is a block address optionally followed by a display format specification. A field modification subcommand is similar to the display subcommand but may include a subfield specification, an operator, and a value. An address specification is a number optionally followed by a type specifier and subfield specification.

The display subcommands are:

<i>num</i>	Display data at absolute address <i>num</i> .
<i>i-numberi</i>	Display data at <i>i-number</i> .
<i>block-addressb</i>	Display data at <i>block-address</i> .
<i>directory-slot-offsetd</i>	Display data at <i>directory-slot-offset</i> .
<b>q</b>	Quit.
<b>!</b>	Escape to the shell.

The display formats are:

<b>p</b>	General display facilities
<b>f</b>	File display facility.

You can step through the i-node information examining each byte, word, or double word. Select the desired display mode by entering one of the following subcommands:

<b>B</b>	Begin displaying in byte mode.
<b>D</b>	Begin displaying in double word mode.
<b>W</b>	Begin displaying in word mode.
<b>O</b>	Toggle error checking on or off.

Moving forward or backward through the i-node data is done with the following symbols:

<b>+num</b>	Move forward the specified number of units currently in effect.
<b>-num</b>	Move backward the specified number of units currently in effect.

The following symbols allow you to store the current address and return to it conveniently:

<b>&gt; address</b>	Store <i>address</i> for later reference. If you do not specify <i>address</i> , <b>fsdb</b> stores the current address.
<b>&lt;</b>	Return to the previously stored address.

The display format applied to the information at the selected address is the one currently in effect. You may receive an error message indicating improper alignment if the address you specify does not fall on an even boundary.

The display facilities display a formatted output in various styles. The current address is normalized to an appropriate boundary before display begins. It advances with the displaying and is left at the address of the last item displayed. The output can be ended at any time by pressing INTERRUPT (**Alt-Pause**).

If you enter a number after the **p** symbol, **fsdb** displays that number of entries. A check is made to detect block boundary overflows because logically sequential blocks are generally not physically sequential. If you enter a count of zero, **fsdb** displays all entries to the end of the current block.

The display formats available are:

<b>i</b>	Display as i-nodes.
<b>d</b>	Display as directories.
<b>o</b>	Display as octal words.
<b>e</b>	Display as decimal words.
<b>c</b>	Display as characters.
<b>b</b>	Display as octal bytes.
<b>y</b>	Display as hex bytes.

Use the **f** symbol to display data blocks associated with the current i-node. If you enter a number after **f**, **fsdb** displays that block of the file. Block numbering begins at zero. The desired display subcommand follows the block number, if present, or the **f** symbol. The display facility works for large as well as small files. It checks for special devices and also checks the data are not zero.

You can use dots (.), tabs, and spaces as subcommand delimiters, but they are not necessary. Pressing just the **Enter** key (entering a blank line) increments the current address by the size of the data type last displayed. That is, the address is set to the next byte, word, double word, directory entry or i-node, allowing you to step through a region of a file system. **fsdb** displays information in a format appropriate to the data type. Bytes, words and double words are displayed as an octal address followed by the octal representation of the data at that address and the decimal equivalent enclosed in parentheses. **fsdb** adds a **.B** or **.D** to the end of the address to indicate a display of byte or double word values. It displays directories as a directory slot offset followed by the decimal i-number and the character representation of the entry name. It displays i-nodes with labeled fields describing each element. The environment variables **NLLDATE** and **NLTIME** control the formats of the date and time.

The following mnemonics are used for the names of the fields of an i-node and refer to the current working i-node:

<b>md</b>	Permission mode
<b>ln</b>	Link count
<b>uid</b>	User number
<b>gid</b>	Group number
<b>sz</b>	File size
<b>an</b>	Data block numbers (0 - 12)
<b>at</b>	Access time
<b>mt</b>	Modification time
<b>maj</b>	Major device number
<b>min</b>	Minor device number

The general form for assigning new values is:

*mnemonic operator new-value*

The **fsdb** command modifies the value of the field specified by *mnemonic* according to the *operator* and *new-value*.

Valid operators include:

- = Assign *new-value* to the specified *mnemonic*.
- =+ Increment the *mnemonic* by the specified *new-value*. The default *new-value* is 1.
- =- Decrease the *mnemonic* by the specified *new-value*. The default *new-value* is 1.
- ='' Assign character string *new-value* to the specified *mnemonic*.

## Examples

The following examples show subcommands that you can use after starting **fsdb**.

1. To display an i-node:

```
386i
```

This displays i-number 386 in i-node format. It now becomes the current i-node.

2. To change the link count for the current i-node to 4:

```
ln=4
```

3. To increase the link count of the current i-node by 1:

```
ln+=1
```

4. To display part of the file associated with the current i-node:

```
fc
```

This displays as ASCII text block zero of the file associated with the current i-node.

5. To display entries of a directory:

```
2i.fd
```

This changes the current i-node to the root i-node (i-node 2), then displays the directory entries in the first block associated with that i-node.

6. To go down a level of the directory tree:

```
d5i.fc
```

This changes the current i-node to the one associated with directory entry 5. Then it displays the first block of the file as ASCII text (fc). Directory entries are numbered starting from 0 (zero).



7. To display a block when you know its block number:

```
1b.p0o
```

This displays the superblock (block 1) of file system in octal.

8. To change the i-number of a directory entry:

```
2i.a0b.d7=3
```

This changes the i-number of directory entry 7 in the root directory (2i) to 3. This example also shows how several operations can be combined on one line.

9. To change the file name of a directory entry:

```
d7.nm="chap1.rec"
```

This changes the name field of directory entry 7 to chap1.rec.

10. To display a given block of the file associated with the current i-node:

```
a2b.p0d
```

This displays block 2 of the current i-node as directory entries.

## Related Information

The following command: “**fsck**, **dfsck**” on page 333.

The **fs** and **dir** files and the **environment** miscellaneous facility in *AIX Operating System Technical Reference*.

The “Overview of International Character Support” in *Managing the AIX Operating System*.

---

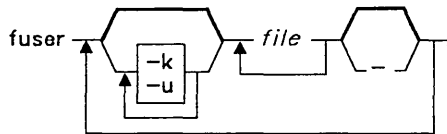
# fuser

---

## Purpose

Identifies processes using a file or file structure.

## Syntax



OL805055

## Description

The **fuser** command lists, for local processes, the process numbers of the processes using the specified local or remote *file*. For remote processes that use local *files*, **fuser** lists the node (NID) that has the files open. It does not list the process numbers, user names, or usage information. For block special devices, all processes using any file on that device are listed. The process number is followed by a letter indicating how the process is using the file:

- c** Using *file* as the current directory
- p** Using *file* as the parent of the current directory (only when in use by the system)
- r** Using *file* as the root directory.

The process numbers are written as a single line to standard output, separated by spaces and ended with a single new-line character. All other output is written to standard error.

## Flags

- k** Sends the **SIGKILL** signal to each local process. Only the person operating with superuser authority can kill another user's process (see "kill" on page 422). **SIGKILL** is not sent to remote processes.
- u** Indicates the login name in parentheses after the process number. The login name is not listed for remote processes.
- Cancels any flags selected for the previous set of file or files.

Flags may be respecified between groups of files on the command line. The new set of flags replaces the old set.

**fuser**

---

**Examples**

1. To list the ID numbers of the processes using the `/etc/passwd` file:

```
fuser /etc/passwd
```

2. To list the process IDs and user names of the processes using the `/etc/filesystems` file:

```
fuser -u /etc/filesystems
```

3. To stop all of the processes using a given disk drive:

```
fuser -k -u /dev/hd1
```

This lists the process ID and user name, and then stops each process that is using the `/dev/hd1` disk drive. You must have superuser authority to stop processes that belong to someone else. You might want to do this if you are trying to unmount `/dev/hd1`, and a process accessing it is preventing you from doing so.

4. To perform the actions of the previous examples in reverse order:

```
fuser -k -u /dev/hd1 - -u /etc/filesystems - /etc/passwd
```

Note that lone dashes before the `-u` and before `/etc/passwd` turn off both the `-k` and `-u` flags.

**Files**

<code>/unix</code>	System kernel image.
<code>/dev/kmem</code>	For system image.
<code>/dev/mem</code>	Also for system image.

**Related Information**

The following commands: “**killall**” on page 425, “**mount**” on page 498, and “**ps**” on page 579.

The **kill** and **signal** system calls in *AIX Operating System Technical Reference*.

“Using Distributed Services” in *Managing the AIX Operating System*.

---

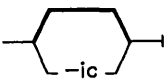

## fwtmp

---

### Purpose

Manipulates connect accounting records.

### Syntax

/usr/lib/acct/fwtmp   
 /usr/lib/acct/wtmpfix   
 /usr/lib/acct/acctwtmp — "reason" —

OL805239

### Description

#### **fwtmp**

The **fwtmp** command reads **wtmp** records from standard input and converts them to formatted ASCII records, which it writes to standard output.

#### **Flag**

**-ic** Reads ASCII input and writes output in binary form.

#### **acctwtmp**

The **acctwtmp** command writes to standard output a **utmp** record containing the string *reason* and the current date and time. *reason* can contain 11 or fewer characters.

### wtmpfix

The **wtmpfix** command examines standard input or the named *files* containing records in **wtmp** format, corrects the date and time stamps to make the entries consistent, and writes the corrected input to standard output. (It is necessary that date and time stamps be consistent because **acctcon1** generates an error and stops when it encounters inconsistent date change records.)

Each time the date is set (on system startup or with the **date** command) a pair of date change records is written to **/usr/adm/wtmp**. The first record is the old date, denoted by the string **old time** placed in the line field and the flag **OLD\_TIME** placed in the type field. The second record is the new date, denoted by the string **new time** placed in the line field and the flag **NEW\_TIME** placed in the type field. The **wtmpfix** command uses these records to synchronize all date and time stamps in the file.

In addition to correcting date and time stamps, **wtmpfix** checks the validity of the name field to ensure that it consists solely of alphanumeric characters, a dollar sign (\$), or spaces. If it encounters an invalid name, it changes the login name to **INVALID** and writes a diagnostic to standard error. In this way, **wtmpfix** reduces the chance that **acctcon2** will fail when it processes connect accounting records.

### Files

**/usr/adm/wtmp**  
**/usr/include/utmp.h**

### Related Information

The following commands: “**acct/\***” on page 31, “**acctcms**” on page 36, “**acctcom**” on page 38, “**acctcon**” on page 42, “**acctdisk**” on page 44, “**acctmerg**” on page 46, “**acctprc**” on page 48, and “**runacct**” on page 606.

The **acct** system call and the **acct** and **utmp** files in *AIX Operating System Technical Reference*.

“Running System Accounting” in *Managing the AIX Operating System*.

---

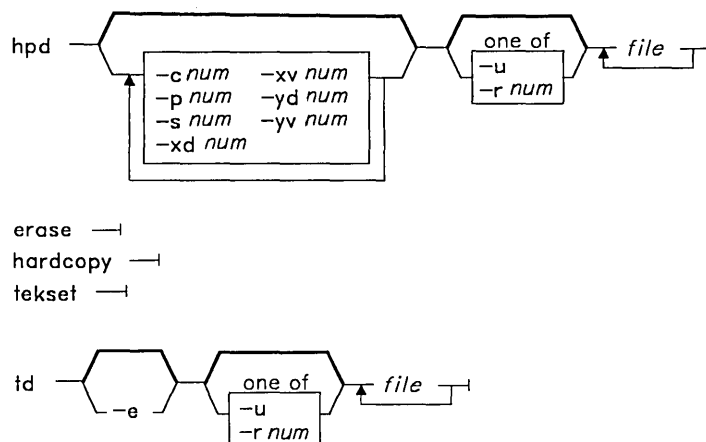
# gdev

---

## Purpose

Provides graphical device routines and filters.

## Syntax



OL777036

## Description

The following commands provide various graphical device routines and filters. They all reside in the `/usr/bin/graf` directory (see “**graphics**” on page 377).

### hpd

The **hpd** command takes a graphical file in **gps** format (standard input by default), and translates it into instructions for the Hewlett-Packard 7221A Graphics Plotter. See the **gps** file in *AIX Operating System Technical Reference* for a description of this file format. It computes a viewing window from the maximum and minimum points in the file, unless you specify the `-r` or `-u` flags.

### **Flags**

- cnum** Selects character set *num*, where *num* is an integer between 0 and 5. See the Hewlett-Packard 7221A Graphics Plotter documentation for a list of these character sets.
- pnum** Selects pen numbered *num*, where *num* is an integer between 1 and 4 inclusive.
- rnum** Displays a window on a **GPS** region, where *num* is an integer from 1 to 25 inclusive.
- snum** Slants characters *num* degrees clockwise from the vertical.
- u** Displays window on the entire **GPS** universe.
- xdnum** Sets **x** displacement of the viewport's lower left corner to *num* inches.
- xvnum** Sets width of viewport to *num* inches.
- ydnum** Sets **y** displacement of the viewport to *num* inches.
- yvnum** Sets height of viewport to *num* inches.

### **erase**

The **erase** command sends characters to a Tektronix 4010 series storage terminal to erase the screen.

### **hardcopy**

When issued at a Tektronix display terminal with a hard copy unit, the **hardcopy** command produces a screen copy on the unit.

### **tekset**

The **tekset** command send characters to a Tektronix terminal to clear the display screen, set the display mode to alpha, and set characters to the smallest font.

### **td**

The **td** command translates a **GPS** object to scope code for a Tektronix 4010 series storage terminal. It computes a viewing window from the maximum and minimum points in *file*, unless you specify the **-u** or **-rnum** flag. Standard input is the default input file.

### **Flags**

- e** Does not erase the screen before initiating display.
- rnum** Displays **GPS** region *num*, where *num* is an integer between 1 and 25 inclusive.
- u** Displays the entire **GPS** universe.

## **Related Information**

The following commands: “**ged**” on page 350, “**gend**” on page 357, and “**graphics**” on page 377.

The **gps** file in *AIX Operating System Technical Reference*.



# ged

---

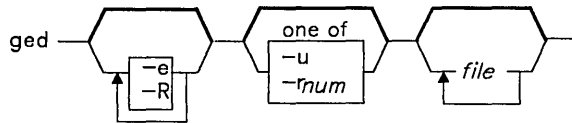
## ged

---

### Purpose

Displays, makes, and edits graphical files on Tektronix 4010 terminals.

### Syntax



OL777037

### Description

The **ged** command is an interactive graphical editor used to edit drawings on Tektronix 4010 series display terminals. The drawings are a sequence of objects that consist of **lines**, **arcs**, and **text**. With **ged** you can view the objects at various magnifications and from various locations. The drawings are stored in graphical primitive string (GPS) files. If you specify - (minus) as the file name, **ged** reads standard input into the edit buffer.

An arc or lines object has a start point (**object-handle**), followed by zero or more points (**point-handles**). A text object has only an object-handle. These objects are positioned within a Cartesian plane (**universe**), having 64K (-32K to +32K) points (**universe-units**) on each axis. The GPS universe is divided into 25 equal sized areas called **regions**. These regions are arranged in five rows of five squares each, numbered 1 to 25 from the lower left of the universe to the upper right.

The **ged** command maps rectangular areas (**windows**) from the universe onto the display screen. Windows let you view pictures from different locations and at different magnifications. The **universe-window** is the window with minimum magnification; that is, the window that views the entire universe. The **home-window** is the window that completely displays the contents of the display buffer.

### Flags

- e Does not erase the screen before the initial display
- rnum Displays region number *num*.
- u Displays the entire GPS universe.
- R Invokes the restricted shell on use of ! (exclamation character).

## Subcommands

The **ged** subcommands are entered in *stages*. Typically each stage ends with a `<cr>` (Return). Prior to the final `<cr>`, you may cancel the subcommand by pressing INTERRUPT (**Alt-Pause**). You can edit the input of a stage, during the stage, by using the erase and kill characters of the calling shell. The `*` (star) prompt indicates that **ged** is waiting at stage 1.

Each subcommand consists of a subset of the following stages:

1. **Command line**, whose format is the same as the format of a shell command:

*subcommand-name* [-flags] [filename]

followed by pressing the **Enter** key. The *subcommand-name* consists of the first character of the subcommand. **ged** echoes the full command name and pauses for the remainder of the command line. Flags are indicated by a leading - (minus). To generate a list of **ged** subcommands, enter: `?`.

2. **Text**, a sequence of characters terminated by an unescaped **Enter**. You can have a maximum of 120 lines of text.
3. **Points**, a sequence of one or more screen locations (maximum of 30), indicated either by the terminal crosshairs or by name. The prompt for entering points is the appearance of the crosshairs. When the crosshairs are visible, typing:

**sp** (space) Enters the current location as a point. The point is identified by a number.

**\$num** Enters the previous point numbered *num*.

**> x** labels the last point entered with the upper case letter *x*.

**\$x** Enters the point labeled *x*.

**.** Establishes the previous points as the current points. At the start of a command, the previous points are those locations given with the previous command.

**=** Echoes the current points.

**\$.num** Enters the point.

**#** Erases the last point entered.

**@** Erases all of the points entered.

4. **Pivot**, a single location entered by pressing the **Enter** key or by using the `$` operator and indicated with a `*` (star).
5. **Destination**, a single location entered by pressing the **Enter** key or by using `$` (dollar sign).

## Subcommand Summary

In the following lists, characters printed in **bold** are to be entered literally. Subcommand stages are printed in ***bold italics***. Arguments surrounded by [] (brackets) are optional. Parentheses surrounding arguments separated by “or” indicate that you must specify exactly one of the arguments.

### Construct Subcommands

**Arc** [-echo,style,weight] ***points***  
**Box** [-echo,style,weight] ***text***  
**Circle** [-echo,style,weight] ***point***  
**Hardware** [-echo] ***text points***  
**Lines** [-echo,style,weight] ***points***  
**Text** [-angle,echo,height, mid-point,right-point,text, weight] ***text points***

### Edit Subcommands

**Delete** (-(universe or view) or ***points***)  
**Edit** [-angle,echo,height,style,weight] (-(universe or view) or ***points***)  
**Kopy** [-echo,points,x] ***points pivot destination***  
**Move** [-echo,points,x] ***points pivot destination***  
**Rotate** [-angle,echo,kopy,x] ***points pivot destination***  
**Scale** [-echo,factor,kopy,x] ***points pivot destination***

### View Subcommands

**coordinates** ***points***  
**erase**  
**new-display**  
**object-handles** (-(universe or view) or ***points***)  
**point-handles** (-(labelled-points or universe or view) or ***points***)  
**view** (-(home or universe or region) or [-x] ***pivot destination***)  
**x** [-view] ***points***  
**zoom** [-out] ***points***

## Other Subcommands

quit or Quit

read [-angle,echo,height, mid-point,right-point,text, weight] *file-name*[*destination*]

set [-angle,echo,factor, height,kopy,mid-point,points, right-point,style,text, weight,**x**]

write *file-name*

!*command*

?

## Options

Options specify parameters used to construct, edit, and view graphical objects. If a parameter used by a subcommand is not specified as an *option*, the default value for the parameter will be used (see set following). The format of subcommand *options* is:

*-option*[,*option*]

where *option* is *keyletter*[*value*]. Flags take on the *values* of true or false indicated by + and - respectively. If no *value* is given with a flag, true is assumed.

## Object Options

<b>anglen</b>	Specifies an angle of <i>n</i> degrees.										
<b>echo</b>	When true, changes made to the display buffer are echoed to the screen.										
<b>factorn</b>	Specifies a scale factor is <i>n</i> percent.										
<b>heightn</b>	Sets the height of text to <i>n</i> universe-units ( $0 \leq n < 1280$ ).										
<b>kopy</b>	When true, copies rather than moves.										
<b>mid-point</b>	When true, uses the mid-point of a text string to locate string.										
<b>points</b>	When true, operates on points; otherwise operates on objects.										
<b>right-point</b>	When true, uses the rightmost point of the text string to locate string.										
<b>styletype</b>	Sets the line style to one of following <i>types</i> : <table> <tr> <td><b>so</b></td> <td>solid</td> </tr> <tr> <td><b>da</b></td> <td>dashed</td> </tr> <tr> <td><b>dd</b></td> <td>dot-dashed</td> </tr> <tr> <td><b>do</b></td> <td>dotted</td> </tr> <tr> <td><b>ld</b></td> <td>long-dashed.</td> </tr> </table>	<b>so</b>	solid	<b>da</b>	dashed	<b>dd</b>	dot-dashed	<b>do</b>	dotted	<b>ld</b>	long-dashed.
<b>so</b>	solid										
<b>da</b>	dashed										
<b>dd</b>	dot-dashed										
<b>do</b>	dotted										
<b>ld</b>	long-dashed.										
<b>text</b>	When false, outlines rather than draws text strings.										
<b>weighttype</b>	Sets line weight to one of following <i>types</i> :										

**n** narrow  
**m** medium  
**b** bold.

## *Area Options*

**home** References the home-window.  
**out** Reduces magnification during zoom.  
**regionn** References the region *n*.  
**universe** References the universe-window.  
**view** References those objects currently in view.  
**x** Indicates the center of the referenced area.

## **Subcommand Descriptions**

### *Construct Subcommands*

#### **Arc**

**Lines** Behave similarly. Each consists of a *command line* followed by *points*. The first *point* entered is the object-handle. Successive *points* are point-handles. **Lines** connects the handles in numerical order. **Arc** fits a curve to the handles (currently a maximum of 3 points will be fit with a circular arc; splines will be added in a later version).

#### **Box**

**Circle** Special cases of **Lines** and **Arc**, respectively. **Box** generates a rectangle with sides parallel to the universe axes. A diagonal of the rectangle would connect the first *point* entered with the last *point*. The first *point* is the object-handle. Point-handles are created at each of the vertices. **Circle** generates a circular arc centered about the *point* numbered zero and passing through the last *point*. The circle's object-handle coincides with the last *point*. A point-handle is generated 180 degrees around the circle from the object-handle.

#### **Text**

**Hardware** Generate *text* objects. Each consists of a *command line*, *text* and *points*. *Text* is a sequence of characters delimited by `<cr>`. Multiple lines of text may be entered by preceding a `cr` with a `\` (backslash). The **Text** subcommand creates software generated characters. Each line of software text is treated as a separate *text* object. The first *point* entered is the object-handle for the first line of text. The **Hardware** command sends the characters in *text*, uninterpreted, to the terminal.

## *Edit Subcommands*

Edit subcommands operate on portions of the display buffer called *defined-areas*. A defined-area is referenced either with an area *option* or interactively. If an area *option* is not given, the perimeter of the defined-area is indicated by *points*. If no *point* is entered, a small defined-area is built around the location of the `<cr>`. This is useful to reference a single *point*. If only one *point* is entered, the location of the `<cr>` is taken in conjunction with the *point* to indicate a diagonal of a rectangle. A defined-area referenced by *points* will be outlined with dotted lines.

- Delete** Removes all objects whose object-handle lies within a defined-area. The **universe** option removes all objects and erases the screen.
- Edit** Modifies the parameters of the objects within a defined-area. Parameters that can be edited are:
- angle** Specifies the angle of *text*
  - height** Specifies the height of *text*
  - style** Specifies the style of *lines* and *arc*
  - weight** Specifies the weight of *lines*, *arc*, and *text*
- Kopy**
- Move** Copies (or moves) object- and/or point-handles within a defined-area by the displacement from the *pivot* to the *destination*.
- Rotate** Rotates objects within a defined-area around the *pivot*. If the **kcopy** flag is true then the objects are copied rather than moved.
- Scale** For object whose object-handles are within a defined-area, point displacements from the *pivot* are scaled by *factor* percent. If the **kopy** flag is true then the objects are copied rather than moved.

## *View Subcommands*

- coordinates** Displays the location of *point(s)* in universe- and screen-units.
- erase** Clears the screen (but not the display buffer).
- new-display** Erases the screen then displays the display buffer.
- object-handles**
- point-handles** Labels object- (and/or point-handles) that lie within the defined-area with **O** (or **P**). **point-handles** identifies labeled points when the **labeled-points** flag is true.
- view** Moves the window so that the universe point corresponding to the *pivot* coincides with the screen point corresponding to the *destination*. Options for **home**, **universe**, and **region** display particular windows in the universe.

## ged

---

- x** Indicates the center of a defined-area. Option view indicates the center of the screen.
- zoom** Decreases (**zoom out**) or increases the magnification of the viewing window based on the defined-area. For increased magnification, the window is set to circumscribe the defined-area. For a decrease in magnification the current window is inscribed within the defined-area.

### *Other Subcommands*

- quit**  
**Quit** Exit from **ged**. **quit** responds with ? if the display buffer has not been written since the last modification.
- read** Inputs the contents of a file. If the file contains a **GPS** object, it is read directly. If the file contains text it is converted into *text* object(s). The first line of a text file begins at *destination*.
- set** When given *option(s)* resets default parameters, otherwise it prints current default values.
- write** Outputs the contents of the display buffer to a file.
- !** Escapes **ged** to execute a AIX Operating System command.
- ?** Lists **ged** subcommands.

### **Related Information**

The following commands: “**gdev**” on page 347, “**graphics**” on page 377, and “**sh**” on page 637.

The **gps** file in *AIX Operating System Technical Reference*.

---

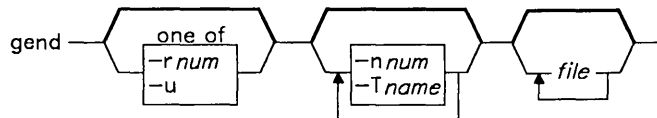
# gend

---

## Purpose

Provides a general graphics device backend.

## Syntax



OL805458

## Description

The `/usr/bin/graf/gend` command displays GPS files on the graphics output devices supported by the Advanced Display Graphics Support Library (GSL). For more information about GSL, see the “Advanced Display Graphics Support Library” in *AIX Operating System Technical Reference*. By default, **gend** reads standard input and writes to the current display (see “**display**” on page 258), but **gend** can also drive printers and plotters if you have installed the VDI drivers that are in the Extended Services Program. You can specify the name of one or more GPS files on the command line. If you enter a file name of - (minus), **gend** reads standard input.

When **gend** displays an image, it opens a new virtual terminal. You can move to and from this virtual terminal by pressing Next Window (**Alt-Action**). See “**open**” on page 541 and *Using the AIX Operating System* for information on virtual terminals. To end **gend** and close the virtual terminal, press END OF FILE (**Ctrl-D**).

**Note:** The **gend** command produces the standard GPS line style attributes with one exception. Line style 4 (long dashed) is rendered as dash-dot-dot.

## Flags

**-nnum** Specifies the number of chords per circle. Legal values are **64**, **128**, **256**, or **512**. The default value is **128**.

Rather than drawing truly circular arcs or circles, **gend** converts them into a series of very short line segments (**chords**), whose end points lie on the circle. For most devices and images, the default value of 128 is satisfactory. The higher values give a smoother image; the lower value provides faster drawing time.



## gend

---

- rnum** Displays data in **GPS** region *num*. A **GPS** object is defined in a Cartesian plane of 64K points on each axis. The plane, or universe, is divided into 25 square regions numbered 1 to 25 from the lower left to the upper right.
- u** Displays data in the entire **GPS** universe.
- Tname** Uses the device specified by the *name* environment variable. The default is the current display (this must be supported by **/dev/hft**). When the image is to be displayed on other devices, you must ensure that the proper VDI device handler is installed.

## Files

- `/usr/bin/graf/gend` The general devices backend.
- `/tmp/dev.XXXXXX` Temporary file.

## Related Information

The following commands: “**ged**” on page 350, “**gdev**” on page 347, “**graphics**” on page 377, and “**open**” on page 541.

“Advanced Display Graphics Support Library” in *AIX Operating System Technical Reference*

Installing programs in *Installing and Customizing the AIX Operating System*.

---

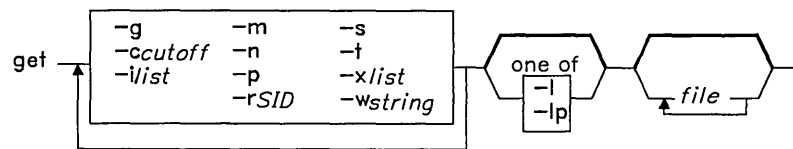
# get

---

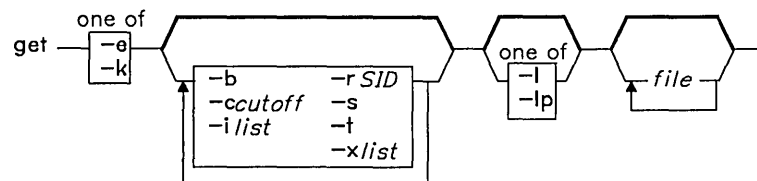
## Purpose

Creates a specified version of a Source Code Control System (SCCS) file.

## Syntax



OL805058



OL805355

## Description

The **get** command reads the specified versions of the named Source Code Control System (**SCCS**) files, creates an ASCII text file for each *file* according to the specified flags, and writes each text file to a file with the same name as the original SCCS file without the *s.* (s period) prefix (the *g-file*). The flags and *files* can be specified in any order, and all flags apply to all named files.

If you specify a directory in place of *file*, **get** performs the requested actions on all the files in the directory that begin with the *s.* prefix. If you specify a - (minus) in place of a *file*, **get** reads standard input and interprets each line as the name of an SCCS file. **get** continues to read input until it reads END OF FILE (**Ctrl-D**).

If the effective user has write permission in the directory containing the SCCS files but the real user does not, then only one file can be named when the **-e** flag is used.

If you are not familiar with the terms *SID* and *delta* or you do not know the numbering system of the deltas, see *AIX Operating System Programming Tools and Interfaces* for more information.

## SCCS Files

In addition to the file with the **s.** prefix (the *s-file*), **get** can create several auxiliary files: the *g-file*, *l-file*, *p-file*, and *z-file*. These files are identified by their *tag*, the letter before the hyphen. **get** names auxiliary files by replacing the leading **s.** in the SCCS file name with the proper tag, except for the *g-file*, which is named by removing the **s.** prefix. So, for a file named **s.sample**, the auxiliary file names would be **sample**, **l.sample**, **p.sample**, and **z.sample**.

These files serve the following purposes:

**s-file** This file contains the original file text and all the changes (*deltas*) made to the file. It also includes information about who can change the file contents, who has made changes, when those changes were made, and what the changes were. You cannot edit this file directly since the file is read-only. It contains the information needed by the SCCS commands to build the *g-file*, the file you can edit.

**g-file** The *g-file* is an ASCII text file that contains the text of the SCCS file version that you specify with the **-r** flag (or the latest trunk version by default). You can edit this file directly. When you have made all your changes and you want to make a new delta to the file, you can then apply the **delta** command to the file. **get** creates the *g-file* in the current directory.

The **get** command creates a *g-file* whenever it runs, unless the **-g** flag or the **-p** flag is specified. The real user owns it (not the effective user). If you do not specify the **-k** or the **-e** flag, the file is read-only. If the **-k** or the **-e** flag is specified, the owner has write permission for the *g-file*. You must have write permission in the current directory to create a *g-file*.

**l-file** The **get** command creates the *l-file* when the **-l** flag is specified. The *l-file* is a read only file. It contains a table showing which deltas were applied in generating the *g-file*. You must have write permission in the current directory to create an *l-file*. Lines in the *l-file* have the following format:

1. A blank character if the delta was applied; a **\*** appears otherwise.
2. A blank character if the delta was applied or was not applied and ignored; a **\*** appears if the delta was not applied and was not ignored.
3. A code indicating a special reason why the delta was or was not applied:

*Blank* Included or excluded normally.

- I** Included using the **-i** flag.
- X** Excluded using the **-x** flag.
- C** Cut off using the **-c** flag.

4. The SID.
5. The date and time the file was created.
6. The login name of person who created the delta.

Comments and MR data follow on subsequent lines, indented one horizontal tab character. A blank line ends each entry.

For example, for a delta cutoff with the `-c` flag, the entry in the l-file might be:

```
**C 1.3 85/03/13 12:44:16 pat
```

and the entry for the initial delta might be:

```
1.1 85/02/27 15:42:20 pat
date and time created 85/02/27 15:42:20 by pat
```

**p-file** The `get` command creates the p-file when the `-e` or the `-k` flag is specified. The p-file passes information resulting from a `get -e` to a `delta` command. The p-file also prevents a subsequent execution of `get` with a `-e` flag for the same SID until `delta` is run or the joint edit keyletter (`j`) is set in the SCCS file. The `j` keyletter allows several `gets` on the same SID. The p-file is created in the directory containing the SCCS *file*. To create a p-file in the SCCS directory, you must have write permission in that directory. The permission code of the p-file is read-only to all but its owner, and it is owned by the effective user. The p-file contains:

- The current SID
- The SID of new delta to be created
- The user name
- The date and time of the `get`
- The `-i` flag, if it was present
- The `-x` flag, if it was present

The p-file contains an entry with the above information for each pending delta for the file. No two lines have the same new delta SID.

**z-file** The z-file is a lock mechanism against simultaneous updates. The z-file contains the binary process number of the `get` command that created it. It is created in the directory containing the SCCS file and exists only while the `get` command is running.

When you use the `get` command, it displays the SID being accessed and the number of lines created from the SCCS file. If you specify the `-e` flag, the SID of the delta to be made appears after the SID accessed and before the number of lines created. If you specify more than one file, or a directory, or standard input, `get` displays the file name before each file is processed. If you specify the `-i` flag, `get` lists included deltas below the word `Included`. If you specify the `-x` flag, `get` lists excluded deltas below the word `Excluded`.

## Identification Keywords

You can use identification keywords in your files to insert identifying information. These keywords are replaced by their values in the g-file when **get** is invoked without the **-e** or **-k** flag. The following identification keywords can be used in SCCS files:

- %M%** Module name: the value of the **m** flag in the SCCS file
- %I%** The SID (%R%.%L%.%B%.%S%) of the g-file
- %R%** Release
- %L%** Level
- %B%** Branch
- %S%** Sequence
- %D%** Date of the current **get** (YY/MM/DD)
- %H%** Date of the current **get** (MM/DD/YY)
- %T%** Time of the current **get** (HH:MM:SS)
- %E%** Date newest applied delta was created (YY/MM/DD)
- %G%** Date newest applied delta was created (MM/DD/YY)
- %U%** Time newest applied delta was created (HH:MM:SS)
- %Y%** Module type: the value of the **t** flag in the SCCS file
- %F%** SCCS file name
- %P%** Full path name of the SCCS file
- %Q%** The value of the **q** flag in the file
- %C%** The current line number. This keyword is intended for identifying messages output by the program. It is not intended to be used on every line to provide sequence numbers.
  
- %Z%** The 4-character string @(#) recognized by the **what** command
- %W%** A shorthand notation for constructing **what** strings for AIX program files. Its value is the characters and keyletters:  
  
**%W%** = **%Z%****%M%**<horizontal-tab>**%I%**
  
- %A%** Another shorthand notation for constructing **what** strings for non-AIX program files. Its value is the keyletters:  
  
**%A%** = **%Z%****%Y%** **%M%** **%I%****%Z%**

The following table illustrates how **get** determines the SID of the file it retrieves, and what the pending SID is. The column **SID Specified** shows the various ways the SID can be specified with the **-r** flag. The two columns illustrate the various conditions that can exist, including whether or not the **-b** flag is used with the **get -e**. The **SID Retrieved** indicates the SID of the file that makes up the g-file. The **SID of Delta to be Created** column indicates the SID of the version that will be created when **delta** is applied.

SID Specified	-b Used	Other Conditions	SID Retrieved	SID of Delta to be Created
none <sup>1</sup>	no	R defaults to mR <sup>2</sup>	mR.mL	mR.(mL + 1)
none <sup>1</sup>	yes	R defaults to mR	mR.mL	mR.mL.(mB + 1).1
(R)elease	no	R > mR	mR.mL	R.1 <sup>3</sup>
R	no	R = mR	mR.mL	mR.(mL + 1)
R	yes	R > mR	mR.mL	mR.mL.(mB + 1).1
R	yes	R = mR	mR.mL	mR.mL.(mB + 1).1
R	N/A	R < mR and R does not exist	hR.mL <sup>4</sup>	hR.mL.(mB + 1).1
R	N/A	R < mR and R exists	R.mL	R.mL.(mB + 1).1
R.(L)evel	no	No trunk successor	R.L	R.(L + 1)
R.L	yes	No trunk successor	R.L	R.L(mB + 1).1
R.L	N/A	Trunk successor in release $\geq$ R	R.L	R.L.(mB + 1).1
R.L.(B)ranch	no	No branch successor	R.L.B.mS	R.L.B.(mS + 1)
R.L.B	yes	No branch successor	R.L.B.mS	R.L.(mB + 1).1
R.L.B.(S)equence	no	No branch successor	R.L.B.S	R.L.B.(S + 1)
R.L.B.S	yes	No branch successor	R.L.B.S	R.L.(mB + 1).1
R.L.B.S	N/A	Branch successor	R.L.B.S	R.L.(mB + 1).1

<sup>1</sup> Applies only if the **d** (default SID) flag is not present in the file (see “**admin**” on page 51)

<sup>2</sup> The mR indicates the maximum existing release.

<sup>3</sup> Forces creation of the first delta in a new release.

<sup>4</sup> The hR is the highest existing release that is lower than the specified, nonexistent, release R.

Figure 2. SID Determination

## Flags

- b** Specifies that the delta to be created should have an SID in a new branch. The new SID is numbered according to the rules stated in Figure 2. You can use **-b** only with the **-e** flag. It is only necessary when you want to branch from a *leaf delta* (a delta without a successor). Attempting to create a delta at a nonleaf delta automatically results in a branch, even if the **b** header flag is not set. If you do not specify the **b** header flag in the SCCS file, **get** ignores the **-b** flag

because the file does not allow branching (see the discussion of header flags on page 54).

**-ccutoff** Specifies a *cutoff* date and time, in the form: *YY[MM[DD[HH[MM[SS]]]]]* **get** includes no deltas to the SCCS file created after the specified *cutoff* in the g-file. The values of any unspecified items in the *cutoff* default to their maximum allowable values. Thus, a cutoff date and time specified with only the year (*YY*) would specify the last month, day, hour, minute, and second of that year. Any number of nonnumeric characters can separate the two-digit items of the *cutoff* date and time. This allows you to specify a date and time in a number of ways, as follows:

```
-c85/9/2,9:00:00
-c"85/9/2 9:00:00"
"-c85/9/2 9:00:00"
```

**-e** Indicates that the g-file being created is to be edited by the user applying **get**. The changes are recorded later with the **delta** command. **get -e** creates a p-file that prevents other users from issuing another **get -e** and editing a second g-file on the same SID before **delta** is run. The owner of the file can override this restriction by allowing joint editing on the same SID through the use of the **admin** command with the **-fj** flag. Other users, with permission, can obtain read-only copies by using **get** without the **-e** flag. The **get -e** command enforces SCCS file protection specified via the ceiling, floor, and authorized user list in the SCCS file (see “**admin**” on page 51).

**-g** Suppresses the actual retrieval of text from the SCCS file. Use the **-g** flag primarily to create an l-file or to verify the existence of a particular SID. Do not use it with the **-e** flag.

**-i***list* Specifies a *list* of deltas to be included in the creation of a g-file. The **SID list format** consists of a combination of individual SIDs separated by commas and SID ranges indicated by two SIDs separated by a hyphen. You specify the same SIDs with both the following command lines:

```
get -e -i1.4,1.5,1.6 s.file
get -e -i1.4-1.6 s.file
```

You can specify the SCCS Identification of a delta in any form shown in the **SID Specified** column of Figure 2 on page 362. **get** interprets partial SIDs as shown in the **SID Retrieved** column of the table.

**-k** Suppresses replacement of identification keywords in the g-file by their value (see “Identification Keywords” on page 362). The **-k** flag is implied by the **-e** flag. If you accidentally ruin the g-file created by **get** with an **-e** flag, you can recreate it by reissuing the **get** command with the **-k** flag in place of the **-e** flag.

- 
- l[p]** Writes a delta summary to an l-file. If you specify **-lp**, the delta summary is written to standard output, and **get** does not create the l-file. Use this flag to determine which deltas were used to create the g-file currently in use. See “SCCS Files” on page 360 for the format of the l-file.
- m** Writes before each line of text in the g-file the SID of the delta that inserted the line into the SCCS file. The format is:  
SID tab line of text
- n** Writes the value of the %M% keyword before each line of text in the g-file (see “Identification Keywords” on page 362 for information on keywords). The format is the value of %M%, followed by a horizontal tab, followed by the text line. When both the **-m** and **-n** flags are used, the format is:  
%M% value tab SID tab line of text
- p** Writes the text created from the SCCS file to standard output and does not create a g-file. **get** sends output normally sent to standard output to file descriptor 2 instead. If you specify the **-s** flag with the **-p** flag, output normally sent to standard output does not appear anywhere. Do not use **-p** with the **-e** flag.
- rSID** Specifies the SCCS identification string (SID) of the SCCS file version to be created. Figure 2 on page 362 shows what version of a file is created and the SID of the pending delta as functions of the SID specified.
- s** Suppresses all output normally written to standard output. Error messages (written to standard error output), remain unaffected.
- t** Accesses the most recently created delta in a given release or release and level. Without the **-r** flag, **get** accesses the most recent delta regardless of its SID.
- wstring** Substitutes *string* for the %W% keyword in g-files not intended for editing (see “SCCS Files” on page 360 for information on g-files).
- xlist** Excludes a *list* of deltas in the creation of a file. See the **-i** flag for the SID list format on page 364.

## Examples

1. To get an SCCS file for editing:

```
get -e s.prog.c
```

This creates a file named `prog.c` that only you have permission to modify. No one else can use `prog.c` or `s.prog.c` until you use the **delta** command to indicate that you are finished.



## get

---

2. To get an SCCS file for reading:

```
get s.prog.c
```

This creates a file named `prog.c` that anyone can read, but that no one can modify. You can do this before searching files with the **grep** command or before compiling programs that are controlled with SCCS. If you are also using the **make** command to manage the development of a software project, **make** automatically does the **get** before compiling a program.

## Related Information

The following commands: “**admin**” on page 51, “**delta**” on page 236, “**help**” on page 391, “**prs**” on page 574, and “**what**” on page 848.

The **scsfile** file in *AIX Operating System Technical Reference*.

The discussion of SCCS in *AIX Operating System Programming Tools and Interfaces*.

---

## getopt

---

### Purpose

Parses command line flags and parameters.

### Syntax

```
set -- 'getopt -- opstring --$*' 1
```

---

<sup>1</sup>This command is not entered on the command line, but is used in shell procedures.

OL805050

### Description

The **getopt** command is used to break up flags and parameters in command lines for easy parsing by shell procedures and to check for valid flags. *opstring* is a string of recognized flags (see the **getopt** subroutine call in *AIX Operating System Technical Reference*). Extended characters are not permitted. If a letter within *opstring* is followed by a colon, the flag is expected to take a modifying parameter that may or may not be separated from it on the command line by one or more tabs or spaces. If you specify -- as the last flag on a command line processed by **getopt**, **getopt** recognizes it and stops its processing; otherwise, **getopt** creates the terminating --. In either case, **getopt** places it at the end of the flags.

When the output from **getopt** is passed by command substitution to the shell **set** command, **set** resets all of the shell positional parameters (\$1, \$2 . . .) so that each flag is preceded by a - (minus) and occupies its own positional parameter. Each parameter (for example, file names and other parameters) is also parsed into its own positional parameter.

The **getopt** command writes a message to standard error when it encounters a flag not included in *opstring*. The **set** command returns a nonzero value if a flag appears on the command line but is not specified in *opstring*. Consequently, you can test the validity of command flags by testing the value of the shell variable \$? . If it is nonzero, the command line contains an unrecognized flag.

### Example

The following shell procedure is a front end to the **ar** command. It uses **getopt** to separate the flags and parameters, then translates them into the **ar** command syntax and runs **ar** with these flags.

## getopt

---

```
# @(#) lib: Front end to the ar command.
#
# Accepts the following flags:
#   ar flags with "-" prefixes.  See the ar command.
#   -L library    The default library is "libsubs.a".
# Note: "lib -r -b sub1.o -v -l newsub.o" performs
#       "ar rrvl sub1.o libsups.a newsub.o".
#       The ar command DOES interpret this correctly.
set -- `getopt c!svmrua:b:i:dpqtxwL: $*`
if [ $? != 0 ]          # Test for syntax error
then
    exit 2
fi
FLAGS= POSNAME= LIBRARY=libsups.a    # Default library name
while [ $1 != -- ]
do
    case $1 in
        -L)
            LIBRARY=$2
            shift; shift    # Shift past the -L and library name
            ;;
        -a|-b|-i)
            FLAGS=$FLAGS`expr "$1" : "-\(.\)\"`
            POSNAME=$2
            shift; shift    # Shift past the flag and parameter
            ;;
        -* )
            # Strip the "-" from the flag
            FLAGS=$FLAGS`expr "$1" : "-\(.\)\"`
            shift
            ;;
    esac
done
shift                    # Shift past the "--" from getopt
FLAGS=${FLAGS:-vt}      # Default if action not specified
ar $FLAGS $POSNAME $LIBRARY $*
```

If this shell procedure is stored in a file named `lib`, then all of the following commands are equivalent:

```
lib -L mylib.a -v -r -b putfld.o getnam.o getfld.o getaddr.o
lib -Lmylib.a -v -r -bputfld.o getnam.o getfld.o getaddr.o
lib -Lmylib.a -vrbputfld.o getnam.o getfld.o getaddr.o
lib -Lmylib.a -v -rbputfld.o -- getnam.o getfld.o getaddr.o
```

In each of these cases, **getopt** breaks down the command into:

```
-L mylib.a -v -r -b putfld.o -- getnam.o getfld.o getaddr.o
```

The **getopt** command writes to its standard output. Because this command is enclosed in `` (grave accents), the shell takes its standard output and uses it to construct the command:

```
set -- -L mylib.a -v -r -b putfld.o -- getnam.o
getfld.o getaddr.o
```

This is called *command substitution*. For more details, see “Command Substitution” on page 647.

The **set** command (page 656) sets the positional parameters \$1, \$2, \$3 . . . to each of the values -L, mylib.a, -v . . . , respectively.

The shell procedure then uses the positional parameters to construct and run the command:

```
ar vrb putfld.o mylib.a getnam.o getfld.o getaddr.o
```

The **ar** command (page 58) accepts the flags in any order. Therefore, you can specify flags to **lib** in any order, as long as a parameter immediately follows a **-a**, **-b**, **-i**, or **-L** flag, and all the flags come before any file names. This means that:

```
lib -bputfld.o -rv -Lmylib.a getnam.o getfld.o getaddr.o
```

produces the command:

```
ar brv putfld.o mylib.a getnam.o getfld.o getaddr.o
```

which performs the same action as each of the previous commands. See “**test**” on page 750 and “**expr**” on page 317 for more information about these commands.

## Related Information

The following command: “**sh**” on page 637.

The **getopt** subroutine in *AIX Operating System Technical Reference*.

# gettext

---

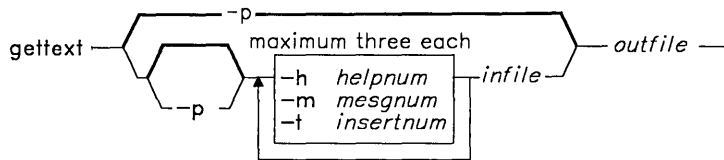
## gettext

---

### Purpose

Extracts message/insert/help descriptions.

### Syntax



OL805130

### Description

The **gettext** command gets message, insert, or help descriptions from *infile* and places the descriptions in *outfile*. If you specify the **-p** flag or **gettext outfile**, **gettext** places a message/insert/help template in *outfile*. When you have your message, insert, or help descriptions or your message/insert/help template in *outfile*; you can edit *outfile*.

The *outfile* is an AIX ASCII file that consists of a header to identify the component and a group of message/insert/help descriptions. The contents of the message/insert/help descriptions includes a delimiter, control information and message/insert/help text. See *AIX Operating System Programming Tools and Interfaces* for a description of the *outfile* format and contents.

### Flags

- h helpnum** Extracts help information from *infile*. You specify the index value used for the desired help number with *helpnum*.
- m mesgnum** Extracts message information from *infile*. You specify the index value used for the desired message number with *mesgnum*.
- p** Makes a message/insert/help template for *outfile*.
- t insertnum** Extracts text insert information from *infile*. You specify the index value used for the desired insert number with *insertnum*.

The syntax for the *mesgnum*, *insertnum*, and *helpnum* parameters is as follows:

- num-num*           Retrieves index numbers *num* to *num*.
- num,num . . .*   Retrieves a list of index numbers specified with *num*, *num*, *num*, and so on (maximum of 50 numbers).
- num-*               Retrieves index numbers equal to and larger than *num*.
- num*               Retrieves index numbers from one to *num*.

## Related Information

The following command: “**puttext**” on page 586.

The discussion of **gettext** in *AIX Operating System Programming Tools and Interfaces*.

# getty

---

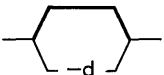

## getty

---

### Purpose

Sets the characteristics of ports.

### Syntax

getty  portname 

OL805333

### Description

The **init** process runs the **getty** command for each *portname* enabled for login. Its primary function is to set the characteristics of the **port** specified by *portname*. Port characteristics include:

- Line speed (baud rate).
- Parity.
- Carriage return, tab, new-line, and form feed delays.
- Character set mapping, such as uppercase to lowercase, carriage return to new-line translation, and tab expansion.
- Extended character support.
- Character erase and line erase editing characters.
- Local or remote echo.
- Screen length for paging.

The **getty** command obtains these settings by reading the port attributes specified in the **/etc/ports** configuration file and by observing the behavior of the port itself. (For details regarding the format of **/etc/ports**, see *AIX Operating System Technical Reference*. For the **logmodes** and **runmodes** parameter settings, see “**stty**” on page 717.) When **getty** is invoked, it first opens the specified port. However, if carrier detection (modem control) is available on the port, **getty** cannot open the port until the carrier is present. Once the port is opened, **getty** sets the work station attributes according to the first **speed**, **logmodes**, **parity**, **erase**, **kill** and other parameters in the **ports** file and writes the herald message **herald** to the port. **getty** then reads a login name from the port. If the login name contains extended characters, they are translated to the single ASCII characters most resembling them.

If a framing error occurs while reading, either because a user generates a **BREAK** signal from the work station or because the line speed is not the same as that of the transmitting work station, the port parameters are reset to the next combination specified in the **ports**

file. Once **getty** reads a login name, it resets the work station modes according to the **runmodes** parameter, turns on carriage-return-to-new-line mapping if the login name was terminated by a carriage return, turns on uppercase-to-lowercase mapping if the alphabetic characters in the login name were uppercase, and executes the program specified by the **logger** parameter. That program, defaulting to **/bin/login**, runs in the same process as **getty** not as its child.

Any additional arguments entered after the login name are passed to the **logger** program. The **login** command interprets these as shell variable settings and places them in the environment.

On dial-in ports, it is often desirable to set no parity generation or checking as a default, but to permit the user to select parity as an option. For example, the following line in the **/etc/ports** file:

```
parity = none,odd+inpck,even+inpck
```

accepts logins with any parity, but if a user generates **BREAK** before typing a login name, **getty** sets the port to generate odd parity and to check incoming characters for odd parity, while two **BREAKs** generate and check for even parity. Similarly, the line:

```
speed=1200,300
```

works with 1200 baud, reverting to 300 baud when a **BREAK** is received before the login name. The default **runmodes** parameter (which must appear on one line in the **ports** file), is generally satisfactory. However, for work stations that have built-in tabs to every eight character positions and do not require tab delays, eliminating the **tab3** from the default in **/etc/ports** will provide faster output with less system load.

### *Special Purpose Options*

If there is a **timeout** keyword in the **ports** file, **getty** waits only the specified number of seconds for a response to the herald before advancing to the next port settings or, after all the settings are exhausted, exiting. If there is a **program** keyword for the port, then instead of displaying the herald and gathering a login name, it executes the specified program immediately. This feature is a general mechanism for supporting special service ports such as network mail demons that need to be spawned when a connection is made from the outside world. As a special case, if you specify:

```
program = HOLD
```

the **runmodes**, **owner**, and **protection** of the port are set and **getty** holds the port open indefinitely, thereby preventing the port modes from reverting to their open-default settings. This is useful, for example, in setting the modes on serial printer ports when it is inconvenient or impossible to have the programs that use them do so.



# getty

---

## Flag

- d** Uses standard input as the work station for which parameters are to be set according to those governing *portname*. Instead of executing a logger or a program, **getty** displays the name of the program that would have been run.

## Example

To test a new **/etc/ports** entry:

```
getty -d /dev/tty5
```

This tests a new port definition for **/dev/tty5** by simulating the login sequence of this device at your work station.

## Files

```
/etc/ports  
/bin/login  
/bin/setmaps
```

## Related Information

The following commands: “**login**” on page 453, “**init**” on page 396, and “**stty**” on page 717.

The **tty** and **ports** files in *AIX Operating System Technical Reference*.

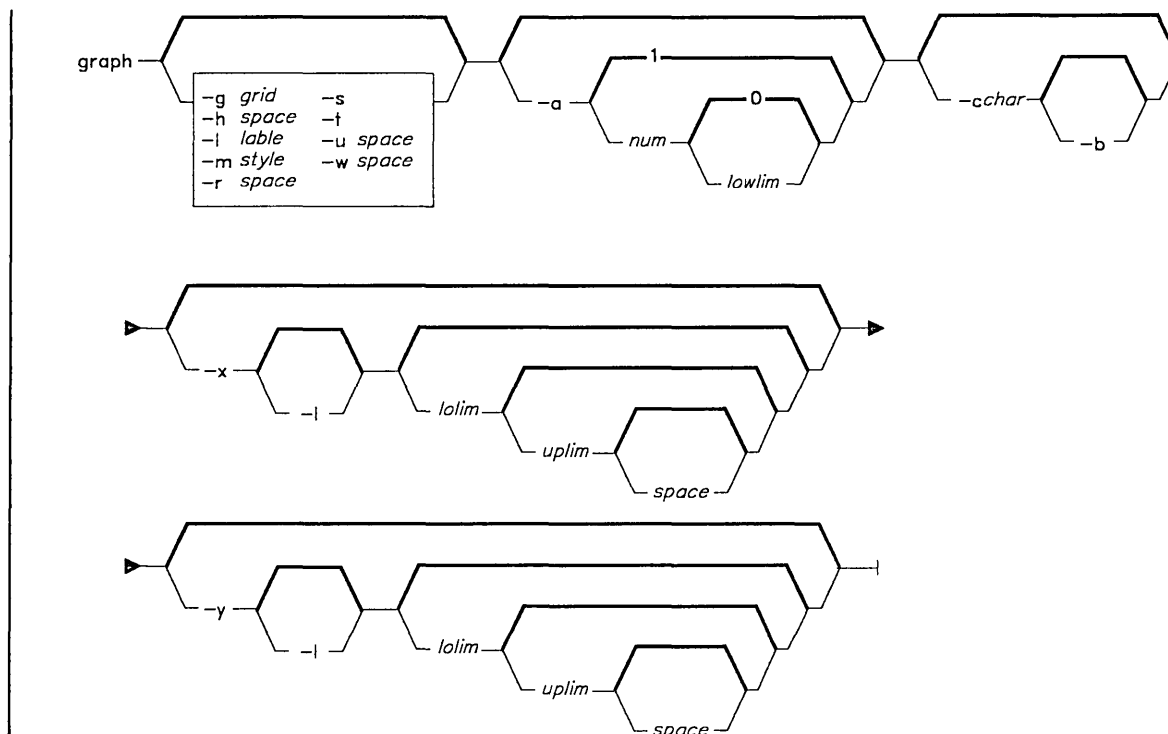
The “Overview of International Character Support” in *Managing the AIX Operating System*.

# graph

## Purpose

Draws a graph.

## Syntax



OL805429

## Description

The **graph** command reads pairs of numbers from standard input, where each pair is the x and y coordinates of a point on a graph. It processes the data so that when it is printed, successive points are connected by straight lines. It writes the graph to standard output. You can then use the **tplot** command to code the output for printing (see “**tplot**” on page 762).

## graph

---

In the input, non-numeric strings following the coordinates of a point are labels. Labels begin on the point. Labels can be surrounded with " (double quotation marks), in which case they can be empty or contain blanks and numbers. Labels cannot contain new-line characters.

The **graph** command stores all points internally and drops those for which there is not room. It also drops segments that run out of bounds.

### Flags

- a** [*num* [*lolim*]]  
Supplies abscissas missing from the input automatically. *num* determines the spacing on the axis (the default is 1). *lolim* determines the starting point for automatic abscissas (the default is 0 or the lower limit given by **-x**[*lolim*]).
- b**  
Breaks the graph after each label in the input.
- c** *char*  
Uses the character string *char* as the default label for each point.
- g** *grid*  
Uses *grid* as the grid style, where *grid* = 0 indicates no grid, *grid* = 1 indicates a frame with tick marks, and *grid* = 2 indicates a full grid (default).
- h** *space*  
Uses *space* as a fraction of space for height.
- l** "*label*"  
Uses *label* as a label for the graph.
- m** *style*  
Uses *style* as the style of connecting lines, where *style*=0 indicates disconnected lines, and *style*=1 indicates connected lines (default).
- r** *space*  
Uses *space* as the fraction of space to move to the right before plotting.
- s**  
Saves the current graphic screen image, does not erase before starting the plot.
- t**  
Transposes horizontal and vertical axes. (**-x** now applies to the vertical axis).
- u** *space*  
Uses *space* as the fraction of space to move up before plotting.
- w** *space*  
Uses *space* as a fraction of space for width.
- x** [*l*] [*lolim* [*uplim* [*space*]]]  
Makes the x axis logarithmic if *l* is used. Use *lolim* as the lower x axis limit and *uplim* as the upper x axis limit. Use *space* for the grid spacing on *x* axis. Normally these are determined automatically.
- y** [*l*] [*lolim* [*uplim* [*space*]]]  
Acts the same as **-x** for the y axis.

The **graph** command produces a legend indicating grid range with a grid unless you specify the **-s** flag. If a specified lower limit exceeds the upper limit, **graph** reverses the axis. Note that logarithmic axes cannot be reversed.

### Related Information

The following commands: "**spline**" on page 684 and "**tplot**" on page 762.

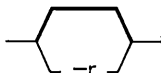
## graphics

---

### Purpose

Accesses graphical and numerical commands.

### Syntax

graphics 

OL777038

### Description

The **graphics** command appends the path name **/usr/bin/graf** to the current **\$PATH** value, changes the primary shell prompt to **^**, and executes a new shell. The directory **/usr/bin/graf** contains all of the graphics subsystem commands.

The command line format for a command in **graphics** is *command name* followed by *argument(s)*. An *argument* may be a *filename* or an *flag string*. A *filename* is the name of any AIX Operating System file, except those beginning with **-**. The *filename -* is the name for the standard input. A *flag string* consists of **-** followed by one or more *flag(s)*. A *flag* consists of a keyletter possibly followed by a value. Flags may be separated by commas.

The graphical commands have been partitioned into four groups.

- Commands that manipulate and plot numerical data; see “**stat**” on page 690.
- Commands that generate tables of contents; see “**toc**” on page 757.
- Commands that interact with graphical devices; see “**gdev**” on page 347 and “**ged**” on page 350.
- A collection of graphical utility commands; see “**gutil**” on page 386.

To produce a list of **graphics** commands, enter **whatis** in the **graphics** environment.

### Flag

- r** Creates access to the graphical commands in a restricted environment; that is, it sets **\$PATH** to **:/usr/bin/graf:rbin:/usr/rbin** and invokes the restricted shell, **rsh**. To restore the environment that existed prior to issuing the **graphics** command, press **Ctrl-D** (END OF FILE). To log out of the graphics environment, enter **quit**.

### Related Information

The following commands: “**gdev**” on page 347, “**ged**” on page 350, “**gend**” on page 357, “**gutil**” on page 386, “**stat**” on page 690, and “**toc**” on page 757.

The **gps** file in *AIX Operating System Technical Reference*.

---

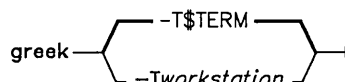
## greek

---

### Purpose

Converts output for a TELETYPE Model 37 work station to output for other work stations.

### Syntax



OL805185

### Description

The **greek** command reinterprets the TELETYPE Model 37 character set, including reverse and half-line motions, for display on other work stations. It simulates special characters, when possible, by overstriking. **greek** reads standard input and writes to standard output.

### Flag

**-Tworkstation** Uses the specified *workstation*. If you omit the **-T** flag, **greek** attempts to use the work station specified in the environment variable **\$TERM** (see the **environ** special facility in *AIX Operating System Technical Reference*.) *workstation* can be any one of the following:

<b>300</b>	DASI 300.
<b>300-12</b>	DASI 300 in 12-pitch.
<b>300s</b>	DASI 300s.
<b>300s-12</b>	DASI 300s in 12-pitch.
<b>450</b>	DASI 450.
<b>450-12</b>	DASI 450 in 12-pitch.
<b>1620</b>	Diablo 1620 (alias DASI 450).
<b>1620-12</b>	Diablo 1620 (alias DASI 450) in 12-pitch.
<b>2621</b>	Hewlett-Packard 2621, 2640, and 2645.
<b>2640</b>	Hewlett-Packard 2621, 2640, and 2645.
<b>2645</b>	Hewlett-Packard 2621, 2640, and 2645.
<b>4014</b>	Tektronix 4014.
<b>hp</b>	Hewlett-Packard 2621, 2640, and 2645.
<b>tek</b>	Tektronix 4014.

## **greek**

---

### **Files**

/usr/bin/300  
/usr/bin/300s  
/usr/bin/4014  
/usr/bin/450  
/usr/bin/hp

### **Related Information**

The following commands: “**300**” on page 863, “**4014**” on page 865, “**450**” on page 866, “**eqn**, **neqn**, **checkeq**” on page 300, “**hp**” on page 392, “**mm**, **checkmm**” on page 492, “**tplot**” on page 762, and “**nroff**” on page 525.

The **greek** miscellaneous facility in *AIX Operating System Technical Reference*.

---

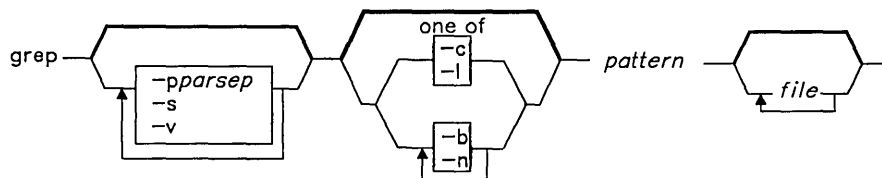
# grep

---

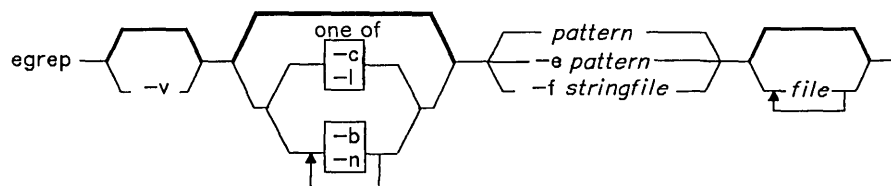
## Purpose

Searches a file for a pattern.

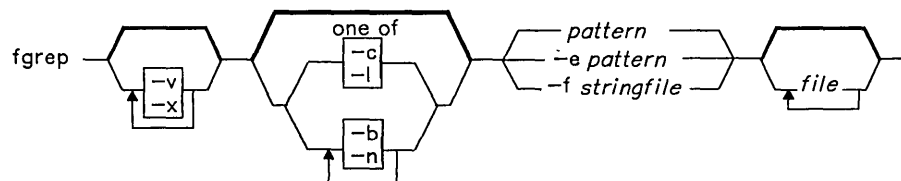
## Syntax



OL805275



OL805359



OL805361

## Description

Commands of the **grep** family search input *files* (standard input by default), for lines matching a pattern. Normally, they copy each line found to standard output. Three versions of the **grep** command permit you to express the matching pattern in varying levels of complexity:

**grep** Searches for *patterns*, which are limited regular expressions in the style of the **ed** command. **grep** uses a compact nondeterministic algorithm.



## grep

---

**egrep** Searches for *patterns* which are full regular expressions as in **ed**, except for \< ( and \) and with the addition of the following rules:

- A regular expression followed by a plus sign (+) matches one or more occurrences of the regular expression.
- A regular expression followed by a question mark (?) matches 0 or 1 occurrences of the regular expression.
- Two regular expressions separated by a vertical bar (|) or by a new-line character match strings that are matched by either.
- A regular expression may be enclosed in parentheses () for grouping.

The order of precedence of operators is [], then \* ? +, then concatenation, then | and the new-line character.

The **egrep** command uses a deterministic algorithm that needs exponential space.

**fgrep** Searches for *patterns* which are fixed strings. It searches for lines that contain one of the strings (lines are separated by new-line characters).

All versions of **grep** display the name of the file containing the matched line if you specify more than one *file* name. Characters with special meaning to the shell (\$ \* [ ! ^ ( ) \), must be quoted when they appear in *patterns*. When *pattern* is not simple string, you usually must enclose the entire *pattern* in single quotation marks. In an expression such as [a-z], the minus means “through” according to the current collating sequence. A collating sequence may define *equivalence classes* for use in character ranges. See the “Overview of International Character Support” in *Managing the AIX Operating System* for more information on collating sequences and equivalence classes.

The exit value of these commands is:

- 0 A match was found.
- 1 No match was found.
- 2 A syntax error was found or a file was inaccessible (even if matches were found).

**Note:** Lines are limited to 512 characters; longer lines are broken into multiple lines of 512 or fewer characters (**grep** only).

Paragraphs (under the **-p** flag) are currently limited to a length of 5000 characters.

Running **grep** on a special file produces unpredictable results and is discouraged.

## Flags

- b** Precedes each line by the block number on which it was found. Use this flag to help find disk block numbers by context.
- c** Displays only a count of matching lines.

- e *pattern*** Specifies a *pattern*. This works the same as a simple *pattern* but is useful when the *pattern* begins with a - (does not work with **grep**).
- f *stringfile*** Specifies a file that contains patterns (**egrep**) or strings (**fgrep**).
- l** Lists just the names of files (once) with matching lines. Each file name is separated by a new-line character.
- n** Precedes each line with its relative line number in the file.
- pparsep** Displays the entire paragraph containing matched lines. Paragraphs are delimited by paragraph separators, *parsep*, which are patterns in the same form as the search pattern. Lines containing the paragraph separators are used only as separators; they are never included in the output. The default paragraph separator is a blank line (**grep** only).
- s** Suppresses error messages about inaccessible files (**grep** only).
- v** Displays all lines except those that match the specified pattern.
- x** Displays lines that match the pattern exactly with no additional characters (**fgrep** only).

## Examples

1. To search several files for a simple string of characters:

```
fgrep "strcpy" *.c
e
```

This searches for the string `strcpy` in all files in the current directory with names ending in `.c`

2. To count the number of lines that match a pattern:

```
fgrep -c "{" pgm.c
fgrep -c "}" pgm.c
```

This displays the number of lines in `pgm.c` that contain open and close braces.

If you do not put more than one `{` or `}` on a line in your C programs, and if the braces are properly balanced, then the two numbers displayed will be the same. If the numbers are not the same, then you can display the lines that contain braces in the order that they occur in the file with: `egrep "{|}" pgm.c`

3. To use a pattern that contains some of the pattern-matching characters `*`, `^`, `?`, `[`, `]`, `\(`, `\)`, `\{`, and `\}`:

```
grep "^[a-zA-Z]" pgm.s
```

This displays all lines in `pgm.s` that begin with a letter.

Note that because **fgrep** does not interpret pattern-matching characters:

```
fgrep "^[a-zA-Z]" pgm.s
```

makes **fgrep** search only for the string `^[a-zA-Z]` in `pgm.s`.

4. To use an extended pattern that contains some of the pattern-matching characters `+`, `?`, `|`, `(`, and `)`:

```
egrep "\\( *[a-zA-Z]*|[0-9]*) *\\" my.txt
```

This displays lines that contain letters in parentheses or digits in parentheses, but not parenthesized letter-digit combinations. It matches `(y)` and `( 783902)`, but not `(alpha19c)`.

**Note:** When using **egrep**, `\(` and `\)` match parentheses in the text, but `(` and `)` are special characters that group parts of the pattern. The reverse is true for **grep**.

5. To display all lines that do *not* match a pattern:

```
grep -v "^#"
```

This displays all lines that do not begin with a `#` character.

6. To display the names of files that contain a pattern:

```
fgrep -l "strcpy" *.c
```

This searches the files in the current directory that end with `.c` and displays the names of those files that contain the string `strcpy`.

## Related Information

The following commands: “**ed**” on page 280, “**sed**” on page 629, and “**sh**” on page 637.

The “Overview of International Character Support” in *Managing the AIX Operating System*.

---

## groups

---

### Purpose

Displays your group membership.

### Syntax

```
groups -[ user1 ]
```

---

<sup>1</sup>The default *user* is the person running the command.

OL805129

### Description

The **groups** command writes to standard output the groups to which you or the specified *user* belong. The AIX Operating System allows you to belong to many different groups at the same time.

Your **primary group** is specified in the file **/etc/passwd**. Once you are logged in, you can change your active group with the **newgrp** command (see page 510). When you create a file, its group ID is that of your active group.

Other groups that you belong to are specified in the file **/etc/group**. If you belong to more than one group, you can access files belonging to any of those groups without changing your primary group ID. These are called your **concurrent groups**.

**Note:** The **/etc/passwd** and **/etc/opasswd** files must be on the same node.

### Files

| **/etc/group**  
| **/etc/passwd**  
| **/etc/opasswd**

### Related Information

The following command: “**newgrp**” on page 510.

The **setgroups** system call and the **initgroups** subroutine in *AIX Operating System Technical Reference*.

# gutil

---

## gutil

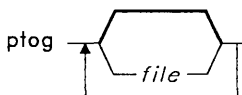
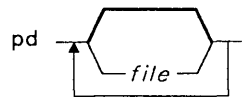
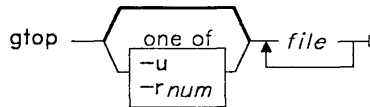
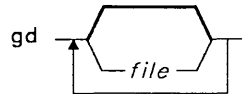
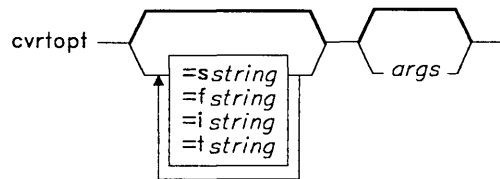
---

### Purpose

Provides graphical utility programs.

### Syntax

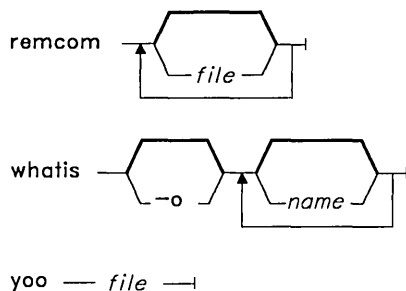
bel  $\rightarrow$



quit  $\rightarrow$

OL777039

OL805449



OL805450

## Description

The following are the miscellaneous device-independent utility commands found in the `/usr/bin/graf` directory. If you do not specify any *files*, these commands read standard input. All output is sent to standard output. Graphical data is stored in GPS format; see the *gps* file in *AIX Operating System Technical Reference*.

### bel

Sends the ASCII BEL character to the terminal.

### cvrtopt

The **cvrtopt** command reformats its arguments (usually the command line arguments of a calling shell procedure) to facilitate processing by shell procedures. An *arg* is either a file name or a flag string. A file name is either a - (minus) by itself or a string not beginning with a -. A flag string is a string of flags beginning with a - (minus). **cvrtopt** produces output of the following form:

```
-flag -flag ... file ...
```

All flags appear singularly in the output and precede any file names. Arguments that take values or are two letters long must be described through flags to **cvrtopt**.

The **cvrtopt** command is usually used with the **set** command as the first line of a shell procedure (see page 656 for a description of the **set** command):

```
set - `cvrtopt [=flags] . . . $@`
```

### Flags

*sstring* The specified *string* accepts string values, where *string* is a one or two letter flag name.

## gutil

---

- fstring* The specified *string* accepts floating point numbers as values, where *string* is a one or two letter flag name.
- istring* The specified *string* accepts integers as values, where *string* is a one or two letter flag name.
- tstring* The specified *string* is a two letter flag name that takes no value.

### gd

The **gd** command produces a readable listing of a file in GPS format.

### gtop

The **gtop** command transforms a GPS format into **plot** file commands displayable by **plot** filters. GPS objects are translated if they fall within the window that circumscribes the first *file*, unless specify one of the following flags:

### Flags

- rnum* Translates objects in GPS region *num*.
- u** Translates all objects in the GPS universe.

### pd

The **pd** command displays a readable listing of **plot** format graphical commands.

### ptog

The **ptog** command transforms **plot** file commands into a GPS file.

### quit

The **quit** command terminates the session.

### remcom

The **remcom** command copies its input to its output with comments removed. Comments are as defined in the C language (*/\* comment \*/*).

### whatis

The **whatis** command displays a short description of each *name* specified. If you do not specify a *name*, then **whatis** displays the current list of description *names*. The command `whatis \*` displays every description.

**Flag**

- o Displays only command flags.

**yoo**

The **yoo** command is a piping primitive that deposits the output of a pipeline into a *file* used in the pipeline. Note that without **yoo**, this is not usually successful because it causes a read and write on the same file simultaneously.

**Related Information**

The following command: “**graphics**” on page 377.

The **gps** format in *AIX Operating System Technical Reference*.



# hangman

---

# hangman

---

## Purpose

Plays hangman, the word-guessing game.

## Syntax

`/usr/games/hangman` 

OL805228

## Description

The **hangman** game chooses a word of at least seven letters from a standard dictionary. You try to guess the word by guessing the letters in it, one at a time. You are allowed seven mistakes. The *file* parameter specifies an alternate dictionary.

To quit the game, press INTERRUPT (**Alt-Pause**) or END OF FILE (**Ctrl-D**).

---

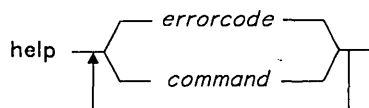
## help

---

### Purpose

Provides information about a Source Code Control System (SCCS) message or command or about certain non-SCCS commands.

### Syntax



OL805054

### Description

The **help** command writes to standard output information about the use of a specified SCCS *command* or about messages generated while using the commands. Each message has an associated *errorcode*, which can be supplied as a argument to the **help** command. Zero or more arguments may be supplied. If you do not supply a argument, **help** prompts for one. You may include any of the SCCS commands as arguments to **help**.

The *errorcode* consists of numbers and letters, and is found at the end of the message. For example, in the message no id keywords (ge6), the error code is ge6.

### Files

/usr/lib/help	Directory containing files of message text.
/usr/lib/help/helploc	File containing locations of help files not in /usr/lib/help.

### Related Information

The discussion of SCCS in *AIX Operating System Programming Tools and Interfaces*.

# hp

---

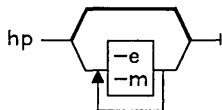
# hp

---

## Purpose

Handles special functions for the HP2640- and HP2621-series terminals.

## Syntax



OL805018

## Description

The **hp** command reads standard input (usually output from **nroff**), and writes to standard output, which is usually Hewlett-Packard 2640-and 2621-series terminal displays. If your terminal has the display enhancement feature, you can display subscripts and superscripts. With the mathematical-symbol feature, you can display Greek and other special characters the same way as the **300** command, with two exceptions: **hp** approximates the logical operator NOT with a right arrow and it only shows the top half of the integral sign.

For overstrike characters (characters followed by a backspace and another character), if either character is an underscore character, the other appears underlined or in inverse video depending on terminal enhancements.

**Note:** Some sequences of control characters (reverse line-feeds and backspaces) can make text disappear from the display. Tables with vertical lines generated by the **tbl** command will often be missing lines of text containing the bottom of a vertical line. You can avoid these problems by first piping the input through **col**, and then through **hp**.

## Flags

- e Shows overstruck characters underlined, superscripts in half-bright, and subscripts half-bright underlined. Otherwise, all overstruck characters, subscripts, and superscripts appear in inverse video (dark-on-light). Use this flag only if your display has the display enhancements feature.
- m Produces only one blank line for any number of successive blank lines in the text.

## Related Information

The following commands: “**300**” on page 863, “**col**” on page 140, “**eqn, neqn, checkedq**” on page 300, “**greek**” on page 379, “**nroff**” on page 525, and “**tbl**” on page 739.

# hyphen

---

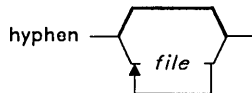
## hyphen

---

### Purpose

Finds hyphenated words.

### Syntax



OL805019

### Description

The **hyphen** command reads the input *files* (standard input by default), finds all the lines ending with hyphenated words, and writes those words to standard output. A word is considered hyphenated only if the hyphen occurs at the end of a line. **hyphen** reads standard input if you do not specify any *file* names on the command line.

**Note:** The **hyphen** command cannot handle hyphenated italic words. It also sometimes gives unnecessary output.

### Examples

1. To check the way words are hyphenated in a text file:

```
hyphen chap1
```

This lists the words in `chap1` that are hyphenated at the end of a line.

2. To check the hyphenation performed by a text formatting program:

```
mm chap1 | hyphen
```

This lists the words that **nroff** decides to hyphenate across lines.

### Related Information

The following commands: “**mm**, **checkmm**” on page 492, “**nroff**” on page 525, and “**troff**” on page 526.

# **id**

---

## **Purpose**

Displays the system identity of the user issuing the command.

## **Syntax**

`id` —

OL805131

## **Description**

The **id** command writes a message to standard output containing the user and group IDs and corresponding names of the invoking process. When effective and real names and IDs do not match, **id** writes both.

## **Related Information**

The following command: “**logname**” on page 456.

The **getuid** subroutine in *AIX Operating System Technical Reference*.

# init

---

# init

---

## Purpose

Initializes the system.

## Syntax

init —<sup>1</sup>

---

<sup>1</sup>This command should not be entered on the command line.

OL805132

## Description

After the kernel completes the basic processor initialization, it starts a process that is the ancestor of all other processes in the system. The process is **init**, the program that controls the state in which the system is running, normally either maintenance mode or multiuser mode. It is the program from which all loggers and most system demons are started.

When **init** starts up, it determines what the startup mode should be based on information in the file **/etc/.init.state**, or, if this file does not exist or is unreadable, on an argument passed to it by the kernel. The usual startup modes for **init** are:

<b>maintenance</b>	Starts a shell on the console, but do not start any other processes (single-user mode).
<b>multiuser</b>	Runs the command file <b>/etc/rc</b> and spawn loggers on all enabled ports.
<b>exec-program</b>	Runs the specified program.

## Maintenance Mode

The maintenance mode is used for system installation, correcting problems on the file system using the **fsck** command, and other operations requiring an inactive system. There are three ways to bring the system up in maintenance mode:

1. If the system is currently running in normal (multiuser) mode, use the **shutdown -m** command to bring the system down to maintenance mode (**shutdown** sends **init** a **SIGINT** signal).

2. Start the system from the Installation/Maintenance Diskette and specify the Maintenance Mode option from the End System Management menu.
3. Edit the file `/etc/.init.state` such that it consists of the character **m**. This causes the system to come up in maintenance mode each time it is started up.

Maintenance mode starts a shell program with superuser authority on the console. When you log off this shell by pressing END OF FILE (**Ctrl-D**), **init** asks you if you want to leave maintenance mode.

A response beginning with **n** or **N** indicates “no,” and **init** starts another shell on the console. Any processes running in the background continue to run. Any other response indicates “yes.”

If the response is yes, **init** enters normal mode, as described in the following section. It also asks if the file system should be assumed to be clean. If you believe this to be true (for example, you have run **fsck** and corrected all problems), answer yes. Your answer determines whether the **rc** command is run with an **m** or **d** argument.

## Normal Mode

After the normal startup of the system (either from system startup or by leaving maintenance mode), **init** runs the normal initialization command file `/etc/rc`. It passes **rc** an argument of either **m** (normal startup, clean root), or **d** (normal startup, dirty root). The latter is the default argument if the startup is from maintenance mode. **rc** is responsible for performing integrity checks, doing any necessary cleanups, mounting the normal file systems, enabling standard ports, and starting system demons. If an error occurs during the running of this command file (indicated by a nonzero return code), **init** either forces a system restart by executing the **reboot** system call or enters maintenance mode.

Once **rc** completes successfully, **init** starts logger processes (normally **getty**) on each enabled port. Whenever someone ends a logger by logging off a port, **init** notes the logout and starts a new logger on the port. Everything **init** knows about enabling ports is contained in the file `/etc/portstatus`, which is maintained by the **penable** command. Through this file, you can enable new ports or disable ports that were previously enabled. Whenever **init** receives a **SIGUP** (hangup) signal, it rereads the **portstatus** file to see if any changes of port status have been requested.

**init** then reads the commands in the `/etc/rc.ds` file, if that file exists. Typically, `/etc/rc.ds` contains commands to start Distributed Services. Any commands that are needed to run remote mounts should be placed in `/etc/rc.ds`.

If, at any time after the system starts up normally, **init** discovers that no ports are enabled or if **init** receives an **INTERRUPT** signal, it decides again on startup options. Generally, this means **init** will go through normal startup, assuming a dirty root.



# init

---

## Environments

Because **init** is the ultimate ancestor of every process on the system, its environment parameters are inherited by every process. As part of its initialization sequence, **init** reads the file **/etc/environment** and copies any assignments found in that file into the environment passed to all of its subprocesses. It treats **umask** differently. If it is assigned a reasonable octal value, **init** does a **umask** system call for the specified value, rather than passing the value in the environment. Similarly, if **filesize** is specified, **init** issues a **ulimit** call with the given size as the argument.

## Files

<b>/etc/utmp</b>	Record of logged-in users.
<b>/usr/adm/wtmp</b>	Permanent login accounting file.
<b>/etc/portstatus</b>	Enabled port status file.
<b>/etc/rc</b>	Initialization command file.
<b>/etc/environment</b>	System-wide environment variables.

## Related Information

The following commands: “**getty**” on page 372, “**penable**” on page 550, “**rc**” on page 594, and “**shutdown**” on page 663

The **reboot** and **umask** system calls and the **portstatus** file in *AIX Operating System Technical Reference*.

The discussion of starting up the system in *Managing the AIX Operating System*.

---

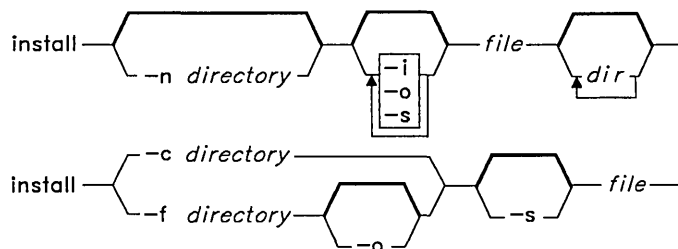
# install

---

## Purpose

Installs a command.

## Syntax



OL805022

## Description

The **install** command installs *file* in a specific place within a file system. It is most often used in “makefiles” (see “**make**” on page 474). When replacing files, **install** copies each file into the appropriate directory, thereby retaining the original owner and permissions. A newly-created *file* has permission code 755, owner **bin**, and group **bin**. **install** writes a message telling you exactly which files it is replacing or creating and where they are going.

If you do not supply any arguments, **install** searches a set of default directories (**/bin**, **/usr/bin**, **/etc**, **/lib**, and **/usr/lib**, in that order) for a file with the same name as *file*. The first time it finds one, it overwrites it with *file* and issues a message indicating that it has done so. If a match is not found, **install** issues a message telling you there was no match and exits with no further action.

If any directories are specified on the command line, **install** searches them before it searches the default directories.

## Flags

**-c directory** Installs a new command file in *directory* only if that file does not already exist there. If it finds a copy of *file* there, it issues a message and exits without overwriting the file. This flag can be used alone or with **-s**.

## install

---

- f** *directory* Forces installation of *file* in *directory* whether or not *file* already exists. If the file being installed does not already exist, **install** sets the permission code and owner of the new file to **755** and **bin**, respectively. This flag can be used alone or with **-o** or **-s**.
- i** Ignores the default directory list and searches only those directories specified on the command line. This flag cannot be used with **-c** or **-f**.
- n** *directory* Installs *file* in *directory* if it is not in any of the searched directories and sets the permissions and owner of the file to **755** and **bin**, respectively. This flag cannot be used with **-c** or **-f**.
- o** Saves the old copy of *file* by copying it to **OLDfile** in the directory in which it found it. This flag cannot be used with **-c**.
- s** Suppresses display of all but error messages.

## Examples

1. To replace a command that already exists in one of the default directories:

```
install fixit
```

This replaces `fixit` if it is found in `/bin`, `/usr/bin`, `/etc`, `/lib`, or `/usr/lib`. Otherwise, it is not installed. For example, if `/usr/bin/fixit` exists, then this file is replaced by a copy of the file `fixit` in the current directory.

2. To replace a command that already exists in a specified or default directory, and to preserve the old version:

```
install -o fixit /etc /usr/games
```

This replaces `fixit` if found in `/etc`, `/usr/games`, or one of the default directories. Otherwise it is not installed. If `fixit` is replaced, the old version is preserved by renaming it **OLDfixit** in the directory in which it was found (**-o**).

3. To replace a command that already exists in a specified directory:

```
install -i fixit /u/jim/bin /u/joan/bin /usr/games
```

This replaces `fixit` if found in `/u/jim/bin`, `/u/joan/bin`, or `/usr/games`. Otherwise it is not installed.

4. To replace a command if found in a default directory, or install it in a specified directory if not found:

```
install -n /usr/bin fixit
```

This replaces `fixit` if found in one of the default directories. If `fixit` is not found, it is installed as `/usr/bin/fixit` (**-n /usr/bin**).

5. To install a new command:

```
install -c /usr/bin fixit
```

This creates a new command by installing a copy of `fixit` as `/usr/bin/fixit`, but only if this file does not already exist.

6. To install a command in a specified directory whether or not it already exists:

```
install -f /usr/bin -o -s fixit
```

This forces `fixit` to be installed as `/usr/bin/fixit` whether or not it already exists. The old version, if any, is preserved by moving it to `/usr/bin/OLDfixit` (-o). The messages that tell where the new command was installed are suppressed (-s).

## Related Information

The following command: “**make**” on page 474.

The **mk** system maintenance procedure in *AIX Operating System Technical Reference*.

# installp

---

## installp

---

### Purpose

Installs a program.

### Syntax

```
installp -d /dev/rfd0 -n $LOGNAME  
installp -d device -n name
```

OL805021

### Description

**Warning:** Before you install a program, you must restart your system and be sure that no other users are on the system and no other programs are running.

The **installp** command installs a program. You must be a member of the system group or operating with superuser authority to run this command.

Because more than one program may be on a set of diskettes, **installp** asks whether or not you want to install each program. If you do, **installp** checks to see if it is an older version than the one currently installed. If it is, **installp** asks if you wish to continue.

The **installp** command makes a backup copy of the program history file before installation begins. If installation is not successful, it sets the Version, Release, and Level fields of the last record of the history file to 00.00.0000 and logs the exit value in the program history file. The history file remains on the system as **/usr/lpp/*pgm-name*/lpp.hist**, where *pgm-name* is the program name.

**Note:** Only ordinary files with the prefix **lpp.** remain in **/usr/lpp/*pgm-name*** after completion of **installp**. All other ordinary files are removed.

You cannot use INTERRUPT (Alt-Pause) to stop the **installp** command. To stop **installp**, press QUIT WITH DUMP (Ctrl-V). This should be used only in extreme circumstances since the state of the system cannot be predicted. For example:

- The write-verify feature may be left on for all minidisks. See “**verify**” on page 830
- All terminals other than the console may be disabled. See “**penable**” on page 550.
- Some install control files may need to be deleted.

---

## Flags

- d *device* Installs the program from the specified *device*. The default *device* is `/dev/rfd0`.
- n *name* Logs the first eight nonblank characters of *name* in the program history file. The default *name* is the value of the environment variable `$LOGNAME`.

The **installp** command runs a program-provided installation procedure **instal**. Each installation procedure returns one of the following exit values to **installp**:

- 0 Installation completed; take no action.
- 2 Update superblocks, i-nodes, and delayed block I/O (sync), then restart the AIX Operating System.
- 3 Build the kernel, then update the superblocks, i-nodes, and delayed block I/O (sync) and shut down the VRM.
- 4 Build the kernel, then update the superblocks, i-nodes, and delayed block I/O (sync) and restart the AIX Operating System.
- 5 Installation cancelled without errors.
- 6 Update superblocks, i-nodes, and delayed block I/O (sync), then shut down the VRM.

Any other return value indicates that installation failed.

## Internal Commands

Install procedures can use the following internal commands. Because they are internal commands, they do a minimum validation of input parameters. Their purpose is to provide common code for the save and recovery functions frequently needed by most program-provided procedures. Because these internal commands function as subcommands, they return exit values rather than issue error messages. However, messages may come from other system commands that they run. C Language programmers of install procedures that call these commands can use the `/usr/include/inu21.h` file to define the return codes for them.

### **inusableave**

The **inusableave** command saves some or all of the files and archive files that will be changed during a program install or update procedure. It uses the following syntax:

```
inusableave listfile pgm-name
```

The *pgm-name* parameter specifies the program to be installed or updated. *pgm-name* can be a maximum of 8 characters. *listfile*, which must be a full path name, contains a list of relative path names (relative to the root) for all of the files that need to be saved. *listfile* must be in the format of an **apply list** (see *AIX Operating System Programming Tools and Interfaces* for a discussion of the format of an apply list).

The **inusableave** command creates the **save directory** (`/usr/lpp/pgm-name/inst_updt.save`). This is the directory in which the install and update procedures store saved files and the

## installp

---

control list that correlates the local file names with their full path names. **inussave** uses *listfile* as a basis to determine which files need to be temporarily saved.

If the file named in *listfile* already exists, **inussave** copies that file to **/usr/lpp/pgm-name/inst\_updt.save/update**. *n*, where *n* is an integer assigned by **inussave**. If the file does not exist, **inussave** assumes that this entry in *listfile* represents either a new file or a file to be archived or processed by the archive procedure. **inussave** maintains a list of saved files in **/usr/lpp/pgm-name/inst\_updt.save/update.list**. The format of each entry in the list is:

**update.n file**

where **update.n** is the name of the saved file and *file* is the full path name of the file.

An archived constituent file is saved if there is a valid archive control file, **/usr/lpp/pgm-name/lpp.acf**, for the program. If this file exists, **inussave** compares each of the file names in *listfile* to the constituent file names in **/usr/lpp/pgm-name/lpp.acf**. When it finds a match, **inussave** uses the **ar** command to extract the constituent file from its associated archive file. It then moves it to **/usr/lpp/pgm-name/inst\_updt.save/archive.n**, where *n* is an integer selected by **inussave**. **inussave** maintains a list of the extracted files that have been saved in the file **/usr/lpp/pgm-name/inst\_updt.save/archive.list**. The format of each entry in the list is:

**archive.n cfile afile**

where **archive.n** is the name of the saved file and *cfile* and *afile* are the constituent and archive files defined in the archive control file.

The **inussave** command returns the following exit values:

- 0** No error conditions occurred.
- 105** Failure occurred trying to create a save directory.
- 107** Copy of a file from one directory to another failed. This implies that the update apply has not yet begun and that the old level of the program is still usable.
- 202** One or more parameters missing.
- 204** Too many parameters were entered.
- 207** Could not access the apply list.

### inurecv

The **inurecv** command recovers all files and archive-constituent files saved from the previous **inussave**. **inurecv** uses the following syntax:

**inurecv pgm-name reject-flag**

It uses the control lists from the **/usr/lpp/pgm-name/inst\_updt.save** directory to recover the files. **inussave** creates the **/usr/lpp/pgm-name/inst\_updt.save** directory and control lists. **inurecv** also recovers files that may have been saved by the program-provided install or update procedure (see *AIX Operating System Programming Tools and Interfaces* for details).

The **inurecv** command has to distinguish between an immediate recovery that occurs because of an error condition during an install or update and an update rejection that occurs because a user rejects an update (`updatep -r`). If the *reject-flag* argument is **yes**, **inurecv** assumes that it is being run because of an update rejection. If the argument is **no** or if no flag is specified, **inurecv** assumes that it is being run because of an immediate recovery.

The **inurecv** command returns the following exit status values:

- 0** No error conditions occurred.
- 101** The save directory does not exist.
- 102** A copy of a file from one directory to another failed. This implies that the program could not be recovered and that it must be reinstalled and any updates reapplied.
- 104** A file that was saved in the save directory was not found.
- 205** Replacement of a constituent file in an archive file failed while attempting to recover a program. This implies that the program is no longer useable and should be reinstalled and any updates reapplied.

### **inurest**

The **inurest** command does simple restores and archives. It does not do any additional processing or user interaction. **inurest** uses the following syntax:

```
inurest [-ddevice] [-q] listfile pgm-name
```

The *listfile* is the full path name of a file containing the relative directory target path name (relative to the root), of files that a program needs to restore. It must be in the format of an apply list. **inurest** restores all files in the list relative to the root directory. *pgm-name* specifies the name of the program to be installed or updated. It can be a maximum of 8 characters.

To archive a file, there must be an archive control file, `/usr/lpp/pgm-name/lpp.acf`. If it exists, **inurest** compares each of the target names in *listfile* to the component files listed in there. Whenever **inurest** finds a match, it archives the restored file into the corresponding archive file and deletes the restored file.

### **Flags**

The following flags modify the action of **inurest**:

- d device** Specifies the input *device*. The default device is `/dev/rfd0`.
- q** Prohibits **restore** from displaying the “insert volume 1” prompt.

The **inurest** command returns the following exit status values:

- 0** No error conditions occurred.
- 106** Failed trying to restore an updated version of files.
- 201** An invalid flag was specified.



**installp**

---

- 202 One or more parameters missing.
- 204 Too many parameters were entered.
- 206 Failed trying to replace file in an archive file.
- 208 Could not access the apply list.

**ckprereq**

The **ckprereq** command determines whether the system level is compatible with the program to be installed or updated. It uses the following syntax:

```
ckprereq [-v] [-f prerequisites]
```

You can run **ckprereq** only if you are a member of the system group or are operating with superuser authority. *prerequisites* is a program prerequisite list file. Each record in this file contains the name of a prerequisite program and describes the version, release, and level requirements. There is one record for each prerequisite program. The default *prerequisites* file is **prereq**. See *AIX Operating System Programming Tools and Interfaces* for details on the format of **ckprereq** file entries.

The **ckprereq** command tests the current version, release, and level found in the history file and marks each “prereq state” field of the **prereq** file with one of the following codes if the test fails:

- n** The **history** file was not found.
- s** There is a syntax error in the **prereq** file.
- v** The test is false for version.
- r** The test is false for release.
- l** The test is false for level.

A blank “prereq state” field indicates that the test was true. The exit value of **ckprereq** is the number of records that did not test true. If all records test true, the exit value is 0.

**Note:** If a program is installed on a local node and executed on a remote node, the remote node must have file trees that have all necessary prerequisite files available.

**Flags**

- f prerequisites** Specifies the *prerequisites* file to use in place of **prereq**.
- v** Sends a descriptive message to standard error for each failure in the prerequisite program test. The messages give the same information as the prereq state field of the **prereq** file.

**mvmd**

The **mvmd** command updates the VRM minidisk. It uses the following syntax:

```
mvmd -a file -D VRM-dir [-fp [file]] -l pgm-name
mvmd -c VRM-file permissions -l pgm-name
mvmd -d VRM-file -l pgm-name
mvmd -m VRM-file [-fp [file]] -l pgm-name
mvmd -r file -D VRM-dir -l pgm-name
```

You must be a member of the system group or operating with superuser authority to run **mvmd**.

**Flags**

- a file** Adds the specified *file* to the VRM minidisk. Use the **-D** flag to specify the destination VRM directory. *file* must not already exist in the specified directory. By default, **mvmd** adds the file to the first unused position in the VRM directory. To specify a position, use the **-f** or **-p** flag.
- c VRM-file permissions** Changes the permission code of the specified *VRM-file* to the octal value, *permissions*. The *VRM-file* parameter must be a full path name. Valid combinations of permission bits are as follows:
- 0700** The loadlist processor loads, runs, and deletes this module.
  - 0450** The loadlist processor transfers control to this module after all loadlist directory entries have been processed.
  - 0440** The loadlist processor loads this module.
  - 0410** This module is a virtual machine.
  - 0040** If the system startup device is a diskette, the loadlist processor is to load the module. If the system startup device is a fixed disk, the loadlist processor does not load the module. Instead, it maps the module.
- The loadlist processor ignores any module that does not have the load bit set. For more information about these permission bits, see *Virtual Resource Manager Technical Reference*.
- d VRM-file** Deletes the specified file from the VRM minidisk. The *VRM-file* parameter must be a full path name.
- D VRM-dir** Specifies the full path name of the VRM directory.
- f [file]** Specifies the position following *file* in the directory list or, if you do not specify *file*, the bottom of the directory list. Use this positioning flag with the **-a** or **-m** flags.

## installp

---

- l *pgm-name*** Specifies the name of a program that is modifying the VRM minidisk. The *pgm-name*, the date, the user name, and a descriptive title are placed in a record appended to the VRM history file. If you do not specify this flag, then a record with the name UNKNOWN is appended to the VRM history file.
- m *VRM-file*** Moves the specified file within its VRM directory. By default, **mvmd** moves the file to the first unused position. To specify a position, use the **-f** or **-p** flag.
- p [*file*]** Specifies the position prior to *file* in the directory list or, if you do not specify *file*, the top of the directory list. Use this positioning flag with the **-a** or **-m** flags.
- r *file*** Replaces the specified *file* on the VRM minidisk. Use the **-D** flag to select the VRM directory of the file to be replaced. Both the replacement file and the file to be replaced must have the same name.

The **mvmd** command returns an exit status of 0 if no errors occurred. A nonzero return indicates that an error did occur.

## Files

<code>instal</code>	Program installation procedure.
<code>liblpp.a</code>	Central archive file.
<code>lpp.hist</code>	Program history file.
<code>prereq</code>	Program prerequisite list file.
<code>/usr/lpp/<i>pgm-name</i>/lpp.acf</code>	Archive control file.
<code>/usr/lpp/<i>pgm-name</i>/inst_updt.save</code>	Directory for saved files.
<code>/usr/lpp/<i>pgm-name</i>/inst_updt/inuPIDtempn</code>	Temporary files.
<code>/usr/include/inu21.h</code>	Defines error codes returned from internal commands.

## Related Information

The following command: “**updatep**” on page 796.

The **fork** and **exec** system calls and the **lpp.hist** file in *AIX Operating System Technical Reference*.

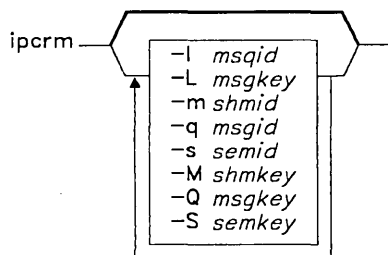
The discussion of installing programs in *AIX Operating System Programming Tools and Interfaces*.

# ipcrm

## Purpose

Removes message queue, semaphore set or shared memory identifiers.

## Syntax



OL805135

## Description

The **ipcrm** command removes one or more message queue, semaphore set, or shared memory identifiers.

## Flags

- lmsgid** Removes local information about the remote queue *msgid* without removing the remote queue.
- Lmsgkey** Removes local information about the remote queue *msgkey* without removing the remote queue.
- mshmkey** Removes the shared memory identifier *shmkey*. The shared memory segment and data structure associated with *shmkey* are also removed after the last detach.
- Mshmkey** Removes the shared memory identifier, created with key *shmkey*. The shared memory segment and data structure associated with it are also removed after the last detach.
- qmsgid** Removes the message queue identifier *msgid* and the message queue and data structure associated with it.

## ipcrm

---

- Q** *msgkey* Removes the message queue identifier, created with key *msgkey*, and the message queue and data structure associated with it.
- s** *semid* Removes the semaphore identifier *semid* and the set of semaphores and data structure associated with it.
- S** *semkey* Removes the semaphore identifier, created with key *semkey*, and the set of semaphores and data structure associated with it.

The details of the remove operations are described in **msgctl**, **shmctl**, and **semctl** in the *AIX Operating System Technical Reference*. The identifiers and keys can be found by using the **ipcs** command.

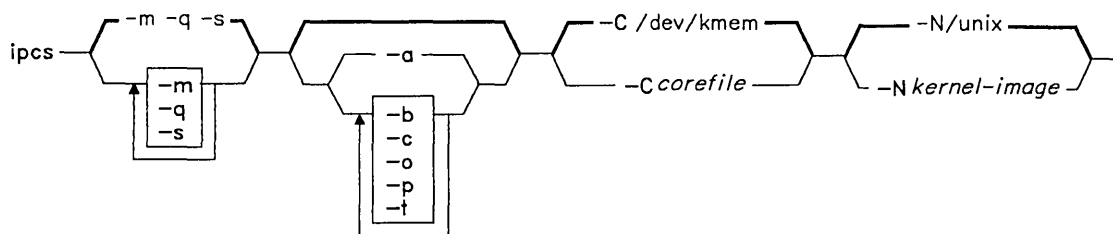
### Related Information

The following command: “**ipcs**” on page 411.

The **msgctl**, **msgget**, **msgrcv**, **msgsnd**, **semctl**, **semget**, **semop**, **shmctl**, **shmget**, and **shmop** system calls in *AIX Operating System Technical Reference*.

**ipcs****Purpose**

Reports inter-process communication facility status.

**Syntax**

OL805432

**Description**

The **ipcs** command writes to the standard output information about active inter-process communication facilities. If you do not specify any flags, **ipcs** writes information in a short form about currently active message queues, shared memory segments, semaphores, remote queues, and local queue headers.

The column headings and the meaning of the columns in an **ipcs** listing follow. The letters in parentheses indicate the flags that cause the corresponding heading to appear. **all** means that the heading always appears. These flags only determine what information is provided for each facility. They do not determine which facilities will be listed.

**T** (all) Type of facility:

- q** message queue
- Q** message queue resides on a remote node
- m** shared memory segment
- s** semaphore.

**ID** (all) The identifier for the facility entry.

**KEY** (all) The key used as a parameter to **msgget**, **semget**, or **shemget** to make the facility entry.

**Note:** The key of a shared memory segment is changed to **IPC\_PRIVATE** when the segment is removed until all processes attached to the segment detach it.

<b>MODE</b>	<p>(all) The facility access modes and flags. The mode consists of 11 characters that are interpreted as follows:</p> <p>The first two characters can be:</p> <ul style="list-style-type: none"><li><b>R</b> if a process is waiting on a <b>msgrcv</b></li><li><b>S</b> if a process is waiting on a <b>msgsnd</b></li><li><b>D</b> if the associated shared memory segment has been removed. It disappears when the last process attached to the segment detaches it.</li><li><b>C</b> if the associated shared memory segment is to be cleared when the first attach is run</li><li>- if the corresponding special flag is not set.</li></ul> <p>The next 9 characters are interpreted as three sets of three bits each. The first set refers to the owner's permissions; the next to permissions of others in the user-group of the facility entry; and the last to all others. Within each set, the first character indicates permission to read, the second character indicates permission to write or alter the facility entry, and the last character is currently unused.</p> <p>The permissions are indicated as follows:</p> <ul style="list-style-type: none"><li><b>r</b> if read permission is granted</li><li><b>w</b> if write permission is granted</li><li><b>a</b> if alter permission is granted</li><li>- if the indicated permission is <i>not</i> granted.</li></ul>
<b>OWNER</b>	(all) The login name of the owner of the facility entry.
<b>GROUP</b>	(all) The name of the group that owns the facility entry.
<b>CREATOR</b>	(a,c) The login name of the creator of the facility entry.
<b>CGROUP</b>	(a,c) The group name of the group of the creator of the facility entry.
	<b>Note:</b> For the <b>OWNER</b> , <b>GROUP</b> , <b>CREATOR</b> , and <b>CGROUP</b> , the user and group IDs display instead of the login names.
<b>CBYTES</b>	(a,o) The number of bytes in messages currently outstanding on the associated message queue.
<b>QNUM</b>	(a,o) The number of messages currently outstanding on the associated message queue.
<b>QBYTES</b>	(a,b) The maximum number of bytes allowed in messages outstanding on the associated message queue.
<b>LSPID</b>	(a,p) The ID of the last process that sent a message to the associated queue. If the last message sent was from a process in a node other than the node which holds the queue, then <b>LSPID</b> is the PID of the kernel process which actually placed the message on the queue, not the PID of the sending process.

<b>LRPID</b>	(a,p) The ID of the last process that received a message from the associated queue. If the last message received was from a process in a node other than the node which holds the queue, then <b>LRPID</b> is the PID of the kernel process which actually received the message on the queue, not the PID of the receiving process.
<b>STIME</b>	(a,t) The time when the last message was sent to the associated queue. For remote queues, this is the server time. No attempt is made to compensate for any clock skew between the local clock and the server clock.
<b>RTIME</b>	(a,t) The time when the last message was received from the associated queue. For remote queues, this is the server time. No attempt is made to compensate for any clock skew between the local clock and the server clock.
<b>CTIME</b>	(a,t) The time when the associated entry was created or changed. For remote queues, this is the server time. No attempt is made to compensate for any clock skew between the local clock and the server clock.
<b>NATTCH</b>	(a,o) The number of processes attached to the associated shared memory segment.
<b>SEGSZ</b>	(a,b) The size of the associated shared memory segment.
<b>CPID</b>	(a,p) The process ID of the creator of the shared memory entry.
<b>LPID</b>	(a,p) The process ID of the last process to attach or detach the shared memory segment.
<b>ATIME</b>	(a,t) The time when the last attach was completed to the associated shared memory segment.
<b>DTIME</b>	(a,t) The time the last detach was completed on the associated shared memory segment.
<b>NSEMS</b>	(a,b) The number of semaphores in the set associated with the semaphore entry.
<b>OTIME</b>	(a,t) The time the last semaphore operation was completed on the set associated with the semaphore entry.

## Flags

<b>-a</b>	Uses the <b>-b</b> , <b>-c</b> , <b>-o</b> , <b>-p</b> and <b>-t</b> flags.
<b>-b</b>	Writes the maximum number of bytes in messages on queue for message queues, the size of segments for shared memory, and the number of semaphores in each semaphores set.
<b>-c</b>	Writes the login name and group name of the user that made the facility.
<b>-Ccorefile</b>	Uses the file <i>corefile</i> in place of <b>/dev/kmem</b> . <i>corefile</i> is a memory image file produced by the <b>Ctrl-(left)Alt-End</b> key sequence.



**ipcs**

---

- m**                   Writes information about active shared memory segments.
- Nkernel-image**   Uses the specified *kernel-image* (**/unix** is the default).
- o**                   Writes the following usage information:
  - Number of messages on queue
  - Total number of bytes in messages in queue for message queues
  - Number of processes attached to shared memory segments.
- p**                   Writes the following:
  - Process number of the last process to receive a message on message queues
  - Process number of the creating process
  - Process number of last process to attach or detach on shared memory segments.
- q**                   Writes information about active message queues.
- s**                   Writes information about active semaphore set.
- t**                   Writes the following:
  - Time of the last control operation that changed the access permissions for all facilities
  - Time of the last *msgsnd* and last *msgrcv* on message queues
  - Time of the last *shmat* and last *shmdt* on shared memory
  - Time of the last *semop* on semaphore sets.

**Files**

<code>/unix</code>	System kernel image.
<code>/dev/kmem</code>	Memory.
<code>/etc/passwd</code>	User names.
<code>/etc/group</code>	Group names.

**Related Information**

The **ipcs**, **msgrcv**, **msgsnd**, **semop**, **shmat**, and **shmdt** system calls in *AIX Operating System Technical Reference*.

The discussion of generating core files in *Problem Determination Guide*.

## ipctable

---

### Purpose

Accesses the Distributed Services IPC Queues Table.

### Syntax

ipctable —

OL805468

### Description

The **ipctable** command lets you build, examine, or change the Distributed Services IPC Queues Table. Only members of the system group or users operating with superuser authority can use **ipctable** to change the state of the Distributed Services IPC Queues Tables (see “su” on page 724). Other user can use **ipctable** to browse the IPC Queues Table.

### Related Information

“Getting Started With Distributed Services Configuration Menus” in *Managing the AIX Operating System*.

## **ipctable**

---

---

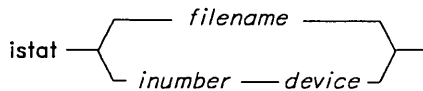
# istat

---

## Purpose

Examines i-nodes.

## Syntax



OL805138

## Description

The **istat** command writes information about the i-nodes specified with *inumber* to standard output. Use the **istat** command to write information about the i-node for a specified *filename*, or to write the contents of a specified i-node, *inumber* on an arbitrary file system.

If you specify *filename*, **istat** writes the following information about the file:

- The device where the file resides.
- The i-node number of the file, on that device.
- The file type (normal, directory, block device, and so on).
- What protection is on the file.
- The name and identification number of the owner and group.

**Note:** The owner and group names for remote files are taken from the local `/etc/passwd` file.

- The number of links to the file.
- If the i-node is for a normal file, the length of the file.
- If i-node is for a device, the major and minor device designations.
- The date of the last time the i-node was updated.
- The date of the last time the file was modified.
- The date of the last time the file was referenced.

If you specify *inumber* and *device*, **istat** also displays, in long decimal values, the block numbers recorded in the i-node. You can specify the *device* as either a device name or as a mounted-file-system name.

**Note:** *inumber* and *device* cannot specify a remote device.

## Examples

1. To display the information stored in a file i-node:

```
istat /bin/sh
```

This displays the i-node information for the file /bin/sh. The information looks something like this:

```
Inode 34 on device 0/10 File
Protection: rwxr-xr-x Sticky
Owner: 0(su) Group: 0(system)
Link count: 1 Length 54240 bytes
```

```
Last updated: Tue Dec 18 01:07:36 1984
Last modified: Sat Jun 30 18:11:47 1984
Last accessed: Wed Feb 13 11:06:37 1985
```

2. To display i-node information if given a file i-number:

```
istat 34 /dev/hd0
```

This displays the information contained in i-node number 34 on the /dev/hd0 device. In addition to the information shown in Example 1, this displays:

```
Block pointers:
  219  220  221  222  223  224  225  226
  227  228  229   0   0
```

These numbers are addresses of the disk blocks that contain the data about the file.

## Related Information

The following command: “**fsdb**” on page 338.

The **stat** system call and the **filesystems** and **fs** files in *AIX Operating System Technical Reference*.

---

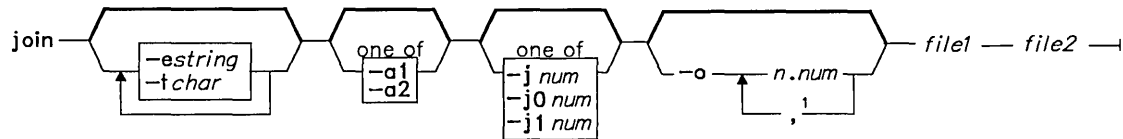
# join

---

## Purpose

Joins data fields of two files.

## Syntax



<sup>1</sup>Do not put a blank on either side of the comma.

OL805371

## Description

The **join** command reads *file1* and *file2*, joins lines in the files according to the flags, and writes the results to standard output. Both files must be sorted according to the collating sequence specified by the **NLCTAB** environment variable, if set, for the fields on which they are to be joined (normally the first field in each line).

One line appears in the output for each identical **join field** appearing in both *file1* and *file2*. The join field is the field in the input files that **join** looks at to determine what will be included in the output. The output line consists of the join field, the rest of the line from *file1*, then the rest of the line from *file2*. You can specify standard input in place of *file1* by substituting a - (minus) for the name.

Both input files must be sorted in increasing ASCII collating sequence on the fields on which they are to be joined (the join field, normally the first field in each line).

Fields are normally separated by a blank, a tab character, or a new-line character. In this case, **join** treats consecutive separators as one, and discards leading separators.

## Flags

**-anum** When *num* is **1**, **join** produces an output line for each line found in *file1* but not in *file2*. When *num* is **2**, **join** produces an output line for each line found in the *file2* but not in *file1*.

## join

---

- e string** Replaces empty output fields with *string*.
- j[n] num** Joins the two files on the *numth* field of file *n*. *n* is **0** or **1**. If you do not specify *n*, **join** uses the *numth* field in each file.
- o n.num[,n.num . . . ]** Makes each output line consist of the fields specified in *list*, in which each element has the form *n.num*, where *n* is a file number and *num* is a field number.
- tchar** Uses *char* as the field separator character in the input and the output. Every appearance of *char* in a line is significant. The default separator is a blank. With default field separation, the collating sequence is that of **sort -b**. If you specify **-t**, the sequence is that of a plain sort. To specify a tab character, enclose it in single quotation marks (").

## Examples

**Note:** The vertical alignment shown in these examples may not be consistent with your output.

1. To perform a simple join operation on two files whose first fields are the same:

```
join phonedir names
```

If phonedir contains the following telephone directory:	and names is this listing of names and department numbers:	then <b>join</b> displays:
Brown J. 555-6235	Elder Dept. 389	Elder G. 555-1234 Dept. 389
Dickerson B. 555-1842	Frost Dept. 217	Green P. 555-2240 Dept. 311
Elder G. 555-1234	Green Dept. 311	McGuff M. 555-5341 Dept. 454
Green P. 555-2240	McGuff Dept. 454	Wilde C. 555-1234 Dept. 520
Harper M. 555-0256	Wilde Dept. 520	
Johnson M. 555-7358		
Lewis B. 555-3237		
McGuff M. 555-5341		
Wilde C. 555-1234		

Each line consists of the join field (the last name), followed by the rest of the line found in `phonedir` and the rest of the line in `names`.

2. To display unmatched lines with the command:

```
join -a2 phonedir names
```

If phonedir contains:	and names contains:	then join displays:
Brown J. 555-6235	Elder Dept. 389	Elder G. 555-1234 Dept. 389
Dickerson B. 555-1842	Frost Dept. 217	Frost Dept. 217
Elder G. 555-1234	Green Dept. 311	Green P. 555-2240 Dept. 311
Green P. 555-2240	McGuff Dept. 454	McGuff M. 555-5341 Dept. 454
Harper M. 555-0256	Wilde Dept. 520	Wilde C. 555-1234 Dept. 520
Johnson M. 555-7358		
Lewis B. 555-3237		
McGuff M. 555-5341		
Wilde C. 555-1234		

This performs the same join operation as in Example 1, and also lists the lines of names that have no match in phonedir. It includes Frost's name and department number in the listing, although there is no entry for Frost in phonedir:

- To display selected fields:

```
join -o 2.3 2.1 1.2 1.3 phonedir names
```

This displays the following fields in the order given:

```
Field 3 of names      (Department Number)
Field 1 of names      (Last Name)
Field 2 of phonedir  (First Initial)
Field 3 of phonedir  (Telephone Number)
```

If phonedir contains:	and names contains:	then join displays:
Brown J. 555-6235	Elder Dept. 389	389 Elder G. 555-1234
Dickerson B. 555-1842	Frost Dept. 217	311 Green P. 555-2240
Elder G. 555-1234	Green Dept. 311	454 McGuff M. 555-5341
Green P. 555-2240	McGuff Dept. 454	520 Wilde C. 555-1234
Harper M. 555-0256	Wilde Dept. 520	
Johnson M. 555-7358		
Lewis B. 555-3237		
McGuff M. 555-5341		
Wilde C. 555-1234		

- To perform the join operation on a field other than the first:

```
sort +2 -3 phonedir ! join -j1 3 - numbers
```

This combines the lines in phonedir and names, comparing the third field of phonedir to the first field of numbers.



## join

---

First, this sorts `phonedir` by the third field, because both files must be sorted by their join fields. The output of `sort` is then piped to `join`. The `-` (minus sign) by itself causes the `join` command to use this output as its first file. The `-j1 3` defines the third field of the sorted `phonedir` as the join field. This is compared to the first field of numbers because its join field is not specified with a `-j` flag.

If numbers contains:	then this command displays the names listed in <code>phonedir</code> for each telephone number:
555-0256	555-0256 Harper M.
555-1234	555-1234 Elder G.
555-5555	555-1234 Wilde C.
555-7358	555-7358 Johnson M.

Note that `join` lists all the matches for a given field. In this case, `join` lists both Elder G. and Wilde C. as having the telephone number 555-1234. The number 555-5555 is not listed because it does not appear in `phonedir`.

## Related Information

The following commands: “`awk`” on page 70, “`comm`” on page 144, “`sort`” on page 672, “`cut`” on page 210, and “`paste`” on page 547.

The **environment** miscellaneous facility in *AIX Operating System Technical Reference*.

The “Overview of International Character Support” in *Managing the AIX Operating System*.

---

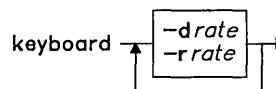
# keyboard

---

## Purpose

Controls the delay and repetition rates of the keyboard.

## Syntax



OL805443

## Description

The **keyboard** command changes the keyboard delay and repetition rates. These rates are initially set at system startup to 500 milliseconds and 14 characters per second, respectively.

## Flags

- d rate** Sets the delay rate to the specified value. *rate* can be **300**, **400**, **500**, or **600** milliseconds.
- rrate** Sets the rate of repetition to the specified value. This *rate* can be an integer from **2** to **40**, inclusive.

## Example

To change both the delay and repetition rates:

```
keyboard -d300 -r40
```

This sets the keyboard to a delay of 300 milliseconds and the repetition rate to 40 characters per second.

# kill

---

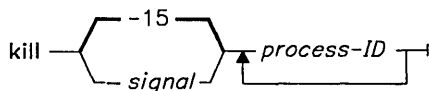
# kill

---

## Purpose

Sends a signal to a running process.

## Syntax



OL805139

## Description

The **kill** command sends a *signal* to a running process, by default signal 15 (**SOFTWARE TERMINATE**). This default action normally kills processes that do not catch or ignore the signal. You specify a process by giving its *process-ID* (process identification number, or **PID**). The shell reports the PID of each process that is running in the background (unless you start more than one process in a pipeline, in which case the shell reports the number of the last process). You can also use the **ps** command to find the process ID number of commands.

In addition, there are special *process-IDs* that cause the following special actions:

- 0** The *signal* is sent to all processes having a process-group ID equal to the process-group ID of the sender (except those with PID's 0 and 1).
- 1** If the effective user ID of the sender is not 0 (root), *signal* is sent to all processes with a process-group ID equal to the effective user ID of the sender. (except those with PID's 0 and 1).  
If the effective user ID of the sender is 0 (root), *signal* is sent to all processes, excluding numbers 0 and 1.
- process-ID** The *signal* is sent to all processes whose process-group number is equal to the absolute value of *process-ID*. Note that when you specify a minus PID, you must also specify the *signal* to be sent, even signal 15.

See the **kill** system call in *AIX Operating System Technical Reference* for a complete discussion of **kill**. For a list of signal numbers, see the **signal** systems call in *AIX Operating System Technical Reference*.

Unless you are operating with superuser authority, the process you wish to stop must belong to you. When operating with superuser authority, you can stop any process.

## Examples

1. To stop a given process:

```
kill 1095
```

This stops process 1095 by sending it the default signal, 15 (also called **SIGTERM**). Note that process 1095 might not actually stop if it has made special arrangements to ignore or override signal 15.

2. To stop several processes that ignore the default signal:

```
kill -9 1034 1095
```

This sends signal 9 (**SIGKILL**) to processes 1034 and 1095. Signal 9 is a special signal that normally cannot be ignored or overridden.

3. To stop all of your background processes:

```
kill 0
```

This sends signal 15 to all members of the shell process group. This includes all background processes started with **&**. (See page 638 about running background processes.) Although the signal is sent to the shell, it has no effect because the shell ignores signal 15.

4. To stop all of your processes and log yourself out:

```
kill -9 0
```

This sends signal 9 to all members of the shell process group. Because the shell cannot ignore signal 9, this also stops the login shell and logs you out. If you are using multiple windows on a high-function terminal, then this closes the active window.

5. To stop all processes that you own:

```
kill -9 -1
```

This sends signal 9 to all processes owned by the effective user, even those started at other work stations and that belong to other process groups. If you are using multiple windows on a high-function terminal, then this closes all of the windows. If a listing that you requested is being printed, then it is also stopped.

**Note:** To send signal 15 with this form of the **kill** command, you must specify **-15** explicitly:

```
kill -15 -1
```

## kill

---

6. To send a different signal code to a process:

```
kill -16 1103
```

This sends signal 16 (SIGUSR1) to process 1103.

The name of the **kill** command is misleading because many signals, including 16, do not stop processes. The action taken on signal 16 is defined by the particular application you are running.

## Related Information

The following commands: “**ps**” on page 579 and “**sh**” on page 637.

The **kill** and **signal** system calls in *AIX Operating System Technical Reference*.

---

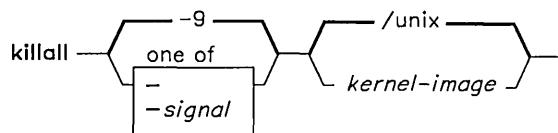
# killall

---

## Purpose

Cancels all processes except the calling process.

## Syntax



OL805140

## Description

The **killall** command cancels all processes that you started, except those producing the **killall** process. This command provides a convenient means of cancelling all processes created by the shell that you control. When started by a user operating with superuser authority, **killall** cancels all cancellable processes except those that started it.

The *kernel-image* parameter specifies the name of the system load module (by default, **/unix**).

## Flags

- Sends a **SIGTERM** signal initially and then sends a **SIGKILL** (kill) signal to all processes that survive for 30 seconds after receipt of the signal first sent. This gives processes that catch **SIGTERM** signal an opportunity to clean up. (For more information, see the **signal** system call in *AIX Operating System Technical Reference*.)
- signal* Sends the specified *signal* number. (For information about signal numbers, see the **signal** system call in *AIX Operating System Technical Reference*.)

## Examples

1. To stop all background processes that have started:

```
killall
```

This sends all background processes the kill signal 9 (also called **SIGKILL**).

## killall

---

2. To stop all background processes, giving them a chance to clean up:

```
killall -
```

This sends signal 15 (**SIGTERM**), waits 30 seconds, and then sends signal 9 (**SIGKILL**).

3. To send a specific signal to the background processes:

```
killall -2
```

This sends signal 2 (**SIGINT**) to the background processes.

## Files

/unix	System kernel image.
/dev/mem	Used for reading the process table.

## Related Information

The following command: “**kill**” on page 422.

The **signal** system call in *AIX Operating System Technical Reference*.

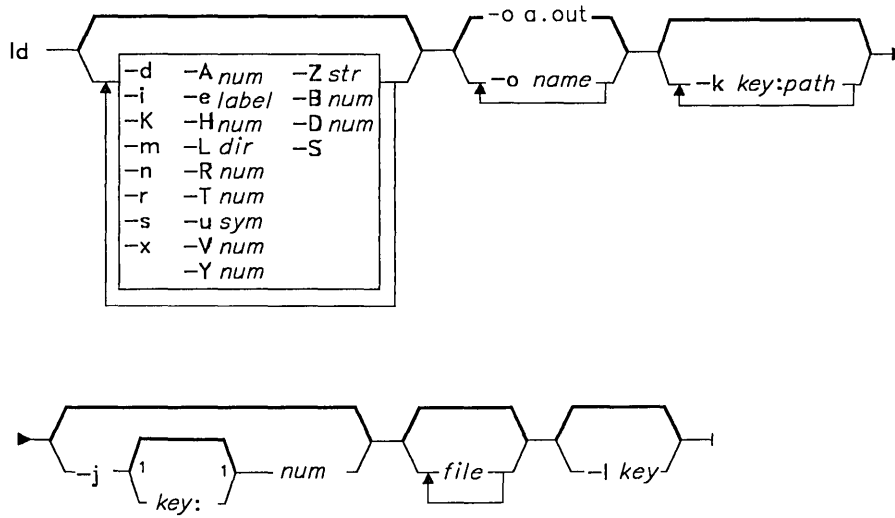
---

**ld**


---

**Purpose**

Links object files.

**Syntax**

OL805362

<sup>1</sup> Do not put a blank between these items.

OL805308

**Description**

The `ld` command (the *linkage editor*) combines the specified object *files* into one, resolving external references and searching libraries. It produces an object module that can be run or that can become a *file* parameter in another call to `ld`. In the latter case, you must use the `-r` flag to preserve the relocation bits. `ld` places its output in a file named `a.out`. It makes this file executable if no errors occur during the link and if the `-r` flag is not specified.

The linkage editor links object files and searches object libraries in the order specified. It links object modules unconditionally, but links from the library only those files that define



an unresolved external reference. If a routine from a library calls another routine in that library, the called routine must follow the calling routine.

Unless you use the `-e` flag to specify another entry point, the first byte of the first nonnull text segment (or the first byte of the data segment if all text segments are null) becomes the entry point of the output file.

The reserved symbols `_text`, `_data`, `_sdata`, `_etext`, `_edata`, and `_end` (in C `text`, `data`, `sdata`, `etext`, `edata`, and `end`) are set to the first location of the program, the first location of the data, the segment number of the data, the first location above the program, the first location above initialized data, and the first location above all data, respectively. You cannot define these symbols.

Because you can use `ld` to link modules intended to run on other machines, some of its action depends upon the architecture of the computer system on which you intend to run the module. `ld` recognizes that architecture automatically from the input modules and modifies its action accordingly. You can use some of its flags to alter the default behavior of `ld` for a particular architecture.

## Flags

The `ld` command recognizes several flags. Except for `-l` entries, which are really abbreviations for file names, the order in which you specify flags does not affect the way they work. You can specify numeric values in either decimal, octal (with a leading `0`), or hexadecimal (with a leading `0x` or `0X`) format.

- `-Anum` Stores *num* in the `a_misc` field of the output file header. This field indicates the size of memory, in bytes, allocated to the process which runs the file. On many systems, the stand-alone loader or kernel uses this value to set the base of the runtime stack pointer.
- `-Bnum` Makes *num* the starting address for the uninitialized data (bss) segment of the output file. The default starting address is the first storage unit after the end of the data segment. Not all architectures support the separation of data and bss segments.
- `-d` Defines common storage, even if you have specified the `-r` flag.
- `-Dnum` Makes *num* the starting address for the initialized data segment of the output file. The default starting address begins at location 0 (if `-i` is in effect), at the first storage unit after the end of the text segment, or, if `-n` is in effect, at the next page or segment boundary.
- `-elabel` Makes *label* the entry point of the executable output file.
- `-Hnum` Makes *num* the boundary, usually the page size, to which the text segment must be padded if it has a different protection than does the data segment. Specify this parameter only to override the default value for the given architecture.

- 
- i** Assigns text and data segments to separate address spaces in memory, with the text segment read-only—if the architecture supports read-only memory—and shared among all users. The data segment starts at location zero unless set with **-D**. If the architecture does not support separate instruction and data space, this flag is treated as if it were **-n**.
- j[key:]num** Assigns the shared library image *key* to location *num*. If you do not specify *key*, do not use location *num* when you assign the runtime location of the shared library text images. The exact interpretation of *num* depends on the target architecture. On the RT PC, *num* refers to the segment register, one of 4 through 13. You can specify **-j** once for each shared library image that has an assigned location.
- kkey:path** Maps any reference to the shared library image with the shared library *key* into *path*. Instead of adding the shared library *key* to the runtime table, add *path*. You can specify **-k** once for each shared library image with a remapped *key*.
- K** Loads the **a.out** header into the first bytes of the text segment, followed by the text segments from the object modules. This flag causes pages of executable files to be aligned on pages in the file system so that they can be demand paged on systems that support paging. This flag provides mapped file support for the text and data segments.
- lkey** Searches the specified library file, where *key* selects the file **libkey.a**. **ld** searches for this file in the directory specified by an **-L** flag, then in **/lib** and **/usr/lib**. It searches library files in the order that you list them on the command line.
- Ldir** Looks in *dir* for files specified by **-I** keys. If it does not find the file in *dir*, **ld** searches the standard directories.
- m** Lists on standard output the names of all files and archive members used to create the output file.
- n** Makes the text segment read-only—if the architecture supports read-only memory—and shared among all users running the file. The data segment starts at the first segment boundary following the end of the text unless set with **-D**. On architectures which only permit read-only text with separate text and data spaces, the **-n** flag is treated as if it were the **-i** flag.
- o name** Assigns *name* rather than **a.out** to the output file.
- r** Writes relocation bits in the output file so that it can serve as a file parameter in another **ld** call. This flag also prevents common symbols from being assigned final definitions and suppresses the undefined symbol diagnostic messages.
- Rnum** Makes *num* bytes the allocation unit for objects manipulated by **ld**, such as segments or common objects. Typically this value ranges from 1 to 8. Specify this parameter only to override the default value for the given architecture.

- s** Strips the symbol table, line number information, and relocation information from the output. This saves space but impairs the usefulness of the debugger. Using the **strip** command has the same effect. This flag is turned off if there are any undefined symbols.
- S num** Makes *num* the maximum size the user stack is allowed to grow. This value represents the number of bytes allowed. If you do not specify this argument, the system assumes a default limit of 1 MB.
- Tnum** Makes *num* the starting address for the text segment of the output file. If not specified, the text segment begins at location zero.
- u sym** Enters *sym* into the symbol table as an undefined symbol. This is useful when linking from only a library, since initially the symbol table is empty and an unresolved reference is needed to force the linking of the first routine.
- Vnum** Stores *num* in the **a\_version** field of the output file header; *num* must be in the range 0 to 32767.
- x** Does not enter local symbols in the output symbol table; enters only external symbols. This flag saves some space in the output file.
- Ynum** In a segmented system, makes *num* the boundary to which the text segment should be padded if it has a protection different from that of the data segment. If *num* is zero, the padding is either that selected by the **-H** flag or the default value associated with that flag. Specify this parameter only to override the default value for the given architecture.
- Zstr** Prefixes with *str* the names specified by the **-l** key. For example, with **-Z/test** and **-lxyz**, **ld** looks for the file **/test/lib/libxyz.a** or, if that file does not exist, **/test/usr/lib/libxyz.a**. The ordinary directories will not be searched. This flag is most useful when cross-compiling.

## Examples

1. To link several object files and produce an **a.out** file to run under the AIX Operating System without the Floating-Point Accelerator:

```
ld -n -t0x10000000 -K /lib/crt0.o pgm.o subs1.o subs2.o -lrts -lc
```

A simpler way to accomplish this is to use the **cc** command to link the files as follows:

```
cc pgm.o subs1.o subs2.o
```

Since the **cc** command automatically uses the link options and necessary support libraries, you do not need to specify them on the command line (it gets this information from the configuration file **cc.cfg**). For this reason, you should use **cc** to link files when you are producing programs that run under the AIX Operating System.

2. To specify the name of the output file:

```
cc -o pgm pgm.o subs1.o subs2.o
```

This stores the linked output in the file `pgm`.

3. To conditionally link library subroutines:

```
cc pgm.o subs1.o subs2.o mylib.a -ltools
```

This links the object modules `pgm.o`, `subs1.o`, and `subs2.o` unconditionally. It then links the subroutines from `mylib.a` that are used by the preceding modules. (This is often called **conditional linking**.) Then `ld` conditionally links subroutines from the library specified by `-ltools`. (This means `/lib/libtools.a`, if it exists. If `ld` does not find this file, then it looks for `/usr/lib/libtools.a`.)

**Note:** Always list libraries and `-l` flags at the end of the `ld` or `cc` command lines.

## Files

<code>/lib/lib*.a</code>	Libraries.
<code>/usr/lib/lib*.a</code>	Libraries.
<code>a.out</code>	Output file.

## Related Information

The following commands: “`ar`” on page 58, “`as`” on page 64, “`cc`” on page 112, and “`shlib`” on page 660.

The `a.out` file in *AIX Operating System Technical Reference*.

The discussion of `ld` in *AIX Operating System Programming Tools and Interfaces* and in *Assembler Language Reference*.

# lex

---

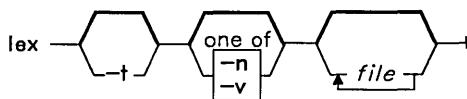
# lex

---

## Purpose

Generates a C Language program that matches patterns for simple lexical analysis of an input stream.

## Syntax



OL805025

## Description

The **lex** command reads *file* or standard input, generates a C Language program, and writes it to a file named **lex.yy.c**. This file, **lex.yy.c**, is a compilable C Language program.

The **lex** command uses *rules* and *actions* contained in *file* to generate a program, **lex.yy.c**, which can be compiled with the **cc** command. It can then receive input, break the input into the logical pieces defined by the *rules* in *file*, and run program fragments contained in the *actions* in *file*. For a more detailed discussion of **lex** and its operation, see *AIX Operating System Programming Tools and Interfaces*.

The generated program is a C Language function called **yylex**. **lex** stores **yylex** in a file named **lex.yy.c**. You can use **yylex** alone to recognize simple, one-word input, or you can use it with other C Language programs to perform more difficult input analysis functions. For example, you can use **lex** to generate a program that simplifies an input stream before sending it to a parser program generated by the **yacc** command.

The function **yylex** analyzes the input stream using a program structure called a “finite state machine.” This structure allows the program to exist in only one state (or condition) at a time. There is a finite number of states allowed. The rules in *file* determine how the program moves from one state to another.

If you do not specify a *file*, **lex** reads standard input. It treats multiple files as a single file.

**Note:** Since **lex** uses fixed names for intermediate and output files, you can have only one **lex**-generated program in a given directory.

## Input File Format (*file*)

The input file can contain three sections; definitions, rules, and user subroutines. Each section must be separated from the others by a line containing only the delimiter, `%%`. The format is:

```

definitions
%%
rules
%%
user subroutines

```

The purpose and format of each are described in the following sections.

### *Definitions*

If you want to use variables in your rules, you must define them in this section. The variables make up the left column, and their definitions make up the right column. For example, if you want to define **D** as a numerical digit, you would write;

```
D [0-9]
```

You can use a defined variable in the rules section by enclosing the variable name in braces (`{D}`).

In the definitions section, you can set table sizes for the resulting finite state machine. The default sizes are large enough for small programs. You may want to set larger sizes for more complex programs.

```

%p n      Number of positions is n (default 2000)
%n n      Number of states is n (default 500)
%t n      Number of parse tree nodes is n (default 1000)
%a n      Number of transitions is n (default 3000)

```

If extended characters appear in regular expression strings, you may need to reset the output array size with the `%o` parameter (possibly to array sizes in the range 10,000 to 20,000). This reset reflects the much larger number of characters relative to the number of ASCII characters.

## Rules

Once you have defined your terms, you can write the rules section. It contains strings and expressions to be matched in *file* to **yylex**, and C commands to execute when a match is made. This section is required, and it must be preceded by the delimiter `%%`, whether or not you have a definitions section. **lex** does not recognize your rules without this delimiter. In this section, the left column contains the pattern to be recognized in an input file to **yylex**. The right column contains the C program fragment executed when that pattern is recognized. Patterns can include extended characters with one exception: these characters may not appear in range specifications within character class expressions surrounded by square brackets. The columns are separated by a tab. For example, if you want to search files for the keyword **KEY**, you might write:

```
(KEY)
printf("found KEY");
```

If you include this rule in *file*, the lexical analyzer **yylex** matches the pattern **KEY** and runs the **printf** command.

Each pattern may have a corresponding action, a C command to execute when the pattern is matched. Each statement must end with a semicolon. If you use more than one statement in an action, you must enclose all of them in braces. A second delimiter, `%%`, must follow the rules section if you have a user subroutine section.

When **yylex** matches a string in the input stream, it copies the matched file to an external character array, **yytext**, before it executes any commands in the rules section.

You can use the following operators to form patterns that you want to match:

- `x` Matches the character written. `x` matches the literal character `x`.
- `[ ]` Matches any one character in the enclosed range (`[.-]`) or the enclosed list (`[...]`). `[a,b,c,x-z]` matches `a,b,c,x,y`, or `z`.
- `" "` Matches the enclosed character or string even if it is an operator. `"$"` prevents **lex** from interpreting the character `$` as an operator.
- `\` Acts the same as `" "`. `\$` also prevents the shell from interpreting the character `$` as an operator.
- `*` Matches zero or more occurrences of the character immediately preceding it. `x*` matches zero or more repeated
- `+` Matches one or more occurrences of the character immediately preceding it.
- `?` Matches either zero or one occurrences of the character immediately preceding it.
- `^` Matches the character only at the beginning of a line. `^x` matches an `x` at the beginning of a line.

---

[^]	Matches any character but the one following the <code>^</code> . <code>[^x]</code> matches any character but <code>x</code> .
.	Matches any character except the new-line character.
\$	Matches the end of a line.
	Matches either of two characters. <code>x   y</code> matches either <code>x</code> or <code>y</code> .
/	Matches one character only when followed by a second character. It reads only the first character into <code>yytext</code> . <code>x/y</code> matches <code>x</code> when it is followed by <code>y</code> , and reads <code>x</code> into <code>yytext</code> .
()	Matches the pattern in the parentheses. This is used for grouping. It reads the whole pattern into <code>yytext</code> . A group in parentheses can be used in place of any single character in any other pattern. <code>(xyz123)</code> matches the pattern <code>xyz123</code> and reads the whole string into <code>yytext</code> .
{}	Matches the character as you defined it in the definitions section. If you defined <code>D</code> to be numerical digits, <code>{D}</code> matches all numerical digits.
{ <i>m,n</i> }	Matches <i>m</i> to <i>n</i> occurrences of the character. <code>x{2,4}</code> matches 2, 3, or 4 occurrences of <code>x</code> .

If a line begins with only a blank, **lex** copies it to the output file, **lex.yy.c**. If the line is in the declarations section of *file*, **lex** copies it to the declarations section of **lex.yy.c**. If the line is in the rules section, **lex** copies it to the program code section of **lex.yy.c**.

### *User Subroutines*

The **lex** library has three subroutines defined as macros, and which you can use in the rules.

<b>input( )</b>	Reads a character from <b>yyin</b> .
<b>unput( )</b>	Replaces a character after it has been read.
<b>output( )</b>	Writes an output character to <b>yyout</b> .

You can override these three macros by writing your own code for these routines in the user subroutines section. But if you write your own, you must undefine these macros in the definition section as follows:

```
%{
#undef input
#undef unput
#undef output
}%
```



## lex

---

There is no **main( )** in **lex.yy.c** because the **lex** library contains the **main( )** that calls **yylex**. Therefore, if you do not include **main( )** in the user subroutines section, when you compile **lex.yy.c**, you must enter **cc -ll lex.yy.c**, where **ll** will call the **lex** library.

External names generated by **lex** all begin with the preface **yy**, as in **yyin**, **yyout**, **yylex**, and **yytext**.

## Flags

- n** Suppresses the statistics summary. When you set your own table sizes for the finite state machine (see page 433), the **lex** automatically produces this summary if you do not select this flag.
- t** Writes **lex.yy.c** to standard output instead of to a file.
- v** Provides a one-line summary of the generated finite-state-machine statistics.

## Files

| /usr/lib/libl.a      Run-time library.

## Related Information

The following command: “**yacc**” on page 861.

The description of **lex** in *AIX Operating System Programming Tools and Interfaces*.

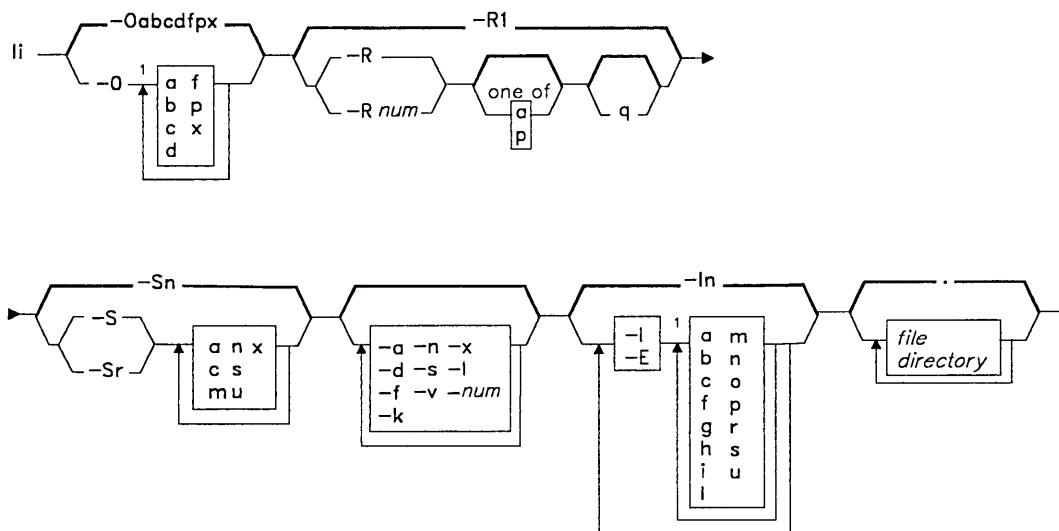
“Overview of International Character Support” in *Managing the AIX Operating System*.

## li

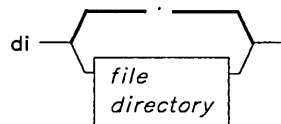
## Purpose

Lists the contents of a directory.

## Syntax



OL805372



OL805346

<sup>1</sup> Do not put a blank between these items.

OL805308

## Description

The **li** command lists the contents of each named *directory* or archive *file* on standard output. For each nonarchive *file* named, **li** displays the file name and any information requested. If you do not specify a *file* or *directory*, **li** lists the current directory.

By default, **li** sorts the output alphabetically and lists it in multiple columns. The collating sequence is determined by the **NLCTAB** environment variable (see “**ctab**” on page 204). It displays control characters in file names in expanded form (for example,  $\wedge D$ ,  $\wedge 177$ , and so on). When you specify more than one file or directory, **li** sorts them appropriately, but files appear before directories and their contents. When the date and time appear, the **NLLDATE** and **NLTIME** environment variables control their format. The **NLSMONTH** environment variable controls the short names for months.

The **di** command is equivalent to **li -lalmops**.

### Permissions Field

The permissions field displayed with the **-lp** flag contains 11 characters. The first character is:

- d** The entry is a directory.
- b** The entry is a block-type special file.
- c** The entry is a character-type special file.
- p** The entry is a pipe (FIFO).
- The entry is an ordinary file.
- D** The entry is a remote directory.
- F** The entry is a remote ordinary file.
- B** The entry is a remote block special file.
- C** The entry is a remote character special file.
- P** The entry is a remote first-in first-out (FIFO) special file.

The next nine characters are interpreted as three sets of three bits each. The first set refers to owner permissions, the next to permissions for others in the same group, and the last to all others. Each of the three characters within each set indicate, respectively, permission to read, write, or execute the file. For a directory, “execute” permission is interpreted as permission to search the directory for a special file. These permissions are indicated as follows:

- r** If the file is readable.
- w** If the file is writable.
- x** If the file is executable.
- If the corresponding permission is not granted.

The group-execute permission is given as **s** if the file has set-group-ID mode. The user-execute permission character is given as **S** if the file has set-user-ID mode. (For a discussion of these modes, see “**chmod**” on page 128.)

The last character of the field is normally blank, but is displayed as **t** if the 1000 bit of the mode is on. (See the **chmod** system call in *AIX Operating System Technical Reference* for the current meaning of this mode.)

**Note:** Some combinations of flags do not work well together. For example, **li -vRa** looks unusual, and **li -RSx** and **li -Sx \*** are both nearly unintelligible if there are subdirectories contained in the current directory, due to confusion about what level is being listed.

## Flags

Flags are grouped into five classes, four of which are always introduced by an uppercase letter: fields (**I** or **E**), restrictions (**O**), recursion (**R**), sort orders (**S**), and miscellaneous. The following flags modify the action of **li**:

**-I [hiplogcsmaunrfb]**

**-E [hiplogcsmaunrfb]**

Requests the inclusion (**-I**) or exclusion (**-E**) of certain fields. These fields are selected by the flags in the subset **hiplogcsmaunrfb**. **-I** includes and **-E** excludes the selected fields in the order in which they appear in the argument list. For example, **-l -Ep** excludes the protections field, while **-Ep -l** includes it, since **-l** (the equivalent of **-l cg| mop**) follows **-Ep**.

The only field included by default is the name (**n**) field. If you include any other fields, **li** lists the output in single-column rather than multiple-column format. **li** lists the following fields in the following order:

**h** Headers  
**i** I-number  
**p** Protections  
**l** Link count  
**o** Local owner (name or UID)  
**g** Local group (name or GID)  
**c** Character count  
**s** Size in blocks  
**m** Modified time  
**a** Accessed time  
**u** Updated (i-node modified) time  
**n** Name  
**r** Node where the entry resides  
**f** Raw UID of the entry's owner  
**b** Raw GID of the entry's group.

If the file is a special file, the size (**s**) field contains the major-and minor-device numbers. If you select the **c** (character count) or **s** (size in blocks) flags, **li** writes a total number of blocks for each directory and a grand total when appropriate.

**li**

For remote files and directories, the local owner and local group are obtained by using inverse IDs. If there is no inverse ID or if **li** cannot determine the inverse ID, a - (minus sign) displays in the corresponding field. If possible, remote nodes are identified with nicknames. Otherwise, they are identified by their NID displayed in hexadecimal. (See “Distributed Services Concepts” in *Managing the AIX Operating System*.)

For local files and directories that do not have a nickname defined for the local node ID, the node ID field displays as a - (minus sign), and the raw UID (GID) field contains the local owner UID (group GID).

**-O [abcdfpx]**

Requests that the listing be restricted to files of certain types. These types are selected from the subset **abcdfpx**. The possible types are:

- a** Archives
- b** Block devices
- c** Character devices
- d** Directories
- f** Files (normal, not special)
- p** Pipes (FIFOs)
- x** Executable files (any file with execute permission)

**-R[num]apq**

Lists recursively to *number* levels deep. The default depth is infinite. This normally displays a single column, with a two-column indentation for each level of the directory structure. When **li** reaches a directory with no subdirectories, it lists the contents of that directory in multiple-column form. Specifying either **-Ra** or **-Rp** suppresses the indentation and multiple-column display. These flags display either the full (**-Ra**) or relative (**-Rp**) path names of each file found. The **-Rq** flag also lists the contents of archive files.

When using the **-Rq** flag to list the contents of remote archive files, the user and group fields display as a - (minus sign) unless the **-k** flag is specified. With the **-k** flag, the user and group fields for archive entries display as raw as found in the archive. (See the archive file format in *AIX Operating System Technical Reference*.)

**-S [acmnrsum]**

Describes the order in which the listing is to be displayed. The default order is by name (**n**). The **-Sx** flag specifies no sorting. Choosing a flag from the subset **acmnsu** selects which field the listing will be sorted by:

- a** Accessed time, latest first
- c** Character count, largest first
- m** Modified time, latest first
- n** Name
- s** Size (same as character count)
- u** Updated time, latest first

If you include the **r** flag with any of these, **li** reverses the order of the sort.

The miscellaneous flags are:

- a** Lists all entries, including those beginning with `.` (dot).
- d** Lists only the name, not the contents, of directories.
- f** Forces `li` to interpret each *file* as a directory and to list the name found in each slot. All flags requiring information not found in directory entries are turned off and the `-a` flag is turned on. Names are listed in the order that they appear in the directory.
- k** Provides a listing that is equivalent to `li -lbcfmpr`. That is, it lists the permission code, node ID, remote UID, remote GID, time of last modification, character count, and file name for remote entries.
- l** Uses a listing that is equivalent to `li -l cglmop` (the long form listing). That is, it lists the permission code, link count, owner, group, character count, time of last modification, time of last access, and name of each file.
- n** Inhibits the interpretation of control characters in file names. This flag is useful for generating lists of file names for program input or for editing into per-file commands.
- s** Provides a listing similar to that of the `-v` flag, except that the distinguishing marks for file types do not affect sorting (a sortable verbose list). Subdirectories appear in the listing as *name/*, files with execute permission as *name\** and special files as *name?*.
- v** Lists files in a way that visually differentiates file types (a verbose visual listing). With this flag, `li` lists subdirectories as [*name*], files with execute permission as `<name>.`, and special files as *\*name\**. This differentiation occurs before the `-S` sort. Thus, different types of files are sorted into different parts of the listing.
- x** Displays every available field except headers (an extended form listing). This is equivalent to specifying `li -labcfglimoprsu`.
- num** Lists with a maximum of *num* columns. If *num* is unreasonable, `li` picks its own *num*. This flag can be used as in `li -l` to make shell files or `li -lO9` to force `li` to display its output in multiple columns. A number appearing in any flag argument is assumed to be the number of columns unless it follows the `-R` flag.

## Examples

1. To list the files in the current directory in alphabetical order:  

```
li
```
2. To list all files in the current directory, including those with names beginning with a `.` (dot):  

```
li -a
```

**li**

---

3. To display detailed information:

```
li -l chap1 .profile
```

This displays a long listing with detailed information about `chap1` and `.profile`. It lists all the information that you probably need to see. However, `li` can supply even more information with the `-x` flag.

4. To display detailed information about a directory:

```
li -d -l . manual manual/chap1
```

This displays a long listing for the directories `.` and `manual`, and for the file `manual/chap1`. `-d` flag, this would list the files in `.` and `manual` instead of the detailed information about the directories themselves.

5. To list the files in order of modification time:

```
li -Sm -l
```

This displays a long listing of the files that were modified most recently, followed by the older files.

6. To include extra information in the listing:

```
li -lchil
```

In addition to the file name, this lists the character count (`-lc`), i-number (`-li`), and link count (`-ll`) for each file in the current directory. The `-lh` tells `li` to write a heading at the top of each column of information.

7. To list the contents of each directory in a tree:

```
li -R manual
```

This lists the names in each subdirectory of the tree that starts with `manual`.

**Files**

`/etc/passwd`      Contains user names for `li -lO`.  
`/etc/group`        Contains group names for `li -lg`.

**Related Information**

The following commands: “`ctab`” on page 204 and “`ls`” on page 461.

The `chmod` system call and the `environment` miscellaneous facility in *AIX Operating System Technical Reference*.

“Overview of International Character Support” in *Managing the AIX Operating System*.

---

“Distributed Services Concepts” in *Managing the AIX Operating System*.





# **line**

---

## **Purpose**

Reads one line from the standard input.

## **Syntax**

line —|

OL805142

## **Description**

The **line** command copies one line from standard input and writes it to standard output. It returns an exit value of 1 on an end-of-file and always writes at least a new-line character. Use this command within a shell command file to read from your work station.

## **Example**

To read a line from the keyboard and append it to a file:

```
echo 'Enter comments for the log:'  
echo ': \c'  
line >>log
```

This shell procedure displays the message:

```
Enter comments for the log:
```

then reads a line of text from the work station keyboard and adds it to the end of log. The `echo ': \c'` command displays a colon prompt. See “**echo**” on page 278 for information about the `\c` escape sequence.

## **Related Information**

The following command: “**sh**” on page 637.

The **read** system call in *AIX Operating System Technical Reference*.

## link

---

## link, unlink

---

### Purpose

Performs the link system call.

### Syntax

```
link - file1 — file2 —|
```

OL805143

```
unlink — file —|
```

OL805227

### Description

The **link** and **unlink** commands perform the corresponding system calls of the same name on the specified file, abandoning all error checking. These commands can be run only by a user operating with superuser authority (see “su” on page 724). You should be familiar with the **link** and **unlink** system calls described in *AIX Operating System Technical Reference*.

The **link** and **unlink** commands do not issue error messages when the associated system call fails; you must check the exit value to determine if the command completed normally. Each returns a 0 if it succeeds, a 1 if you specify too few or too many parameters, and a 2 if its system call fails.

**Warning:** The **link** and **unlink** commands allow the superuser to deal with unusual problems, such as moving an entire directory to a different part of the directory tree. They also permit you to create directories that cannot be reached or escaped from. Be careful to preserve directory structure by observing the following rules:

- Be certain every directory has a . (dot) link to itself.
- Be certain every directory has a .. (dot dot) link to its parent directory.
- Be certain every directory has no more than one link to it.
- Be certain every directory is accessible from the root of its file system.

## Example

To restore the . (dot) entry of the damaged directory dir:

```
link dir dir/.
```

**Warning:** Do this only if the . (dot) entry has somehow been destroyed and **fsck** is unable to repair it. This happens very rarely.

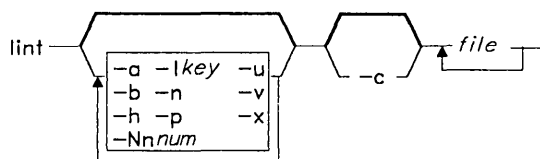
## Related Information

The following commands: “**ln**” on page 450 and “**fsck, dfsc**” on page 333.

The **link** and **unlink** system calls in *AIX Operating System Technical Reference*.

**lint****lint****Purpose**

Checks C programs for potential problems.

**Syntax**

OL805433

**Description**

The **lint** program checks C language source code for coding and syntax errors and for inefficient or nonportable code. You can use this program to

- Identify source code and library incompatibility
- Enforce type checking rules more strictly than does the compiler
- Identify potential problems with variables
- Identify potential problems with functions
- Identify problems with flow control
- Identify legal constructions that may produce errors or be inefficient
- Identify possibly nonportable code.

The **lint** command assumes that *file* names ending in **.c** are C Language source files. It assumes that those ending in **.ln** are the result of an earlier running of **lint** with either the **-c** or the **-o** flag used. These **.ln** files are analogous to the **.o** (object) files produced by the **cc** command when given a **.c** file as input. **lint** warns you about files with other suffixes and ignores them.

The **lint** command takes all the **.c** and **.ln** files and the libraries specified by **-l** flags and processes them in the order that they appear on the command line. By default, it adds the standard **lint** library (**llib-1c.ln**) to the end of the list of files. However, when you select the **-p** flag, **lint** uses the portable library **llib-port.ln**. By default, the second pass of **lint** checks this list of files for mutual compatibility; however, if you specify the **-c** flag, **lint** ignores the **.ln** and **lib-1x** files.

The **-c** and **-o** flags allow for incremental use of **lint** on a set of C Language source files. Generally, you use **lint** once for each source file with the **-c** flag. Each of these runs produces a **.ln** file that corresponds to the **.c** file and writes all messages that are about

just that source file. After you have run all source files separately through **lint**, you run it once more, without the **-c** flag, listing all the **.ln** files with the needed **-l** arguments. This writes all inter-file inconsistencies. This procedure works well with the **make** command, allowing it to run **lint** on only those source files that have been modified since the last time that set of source files was checked.

The following comments in a C source program change the way that **lint** operates when checking the source program:

- /\*NOTREACHED\*/** Suppresses comments about unreachable code.
- /\*VARARGS*n*\*/** Suppresses checking the following function declaration for varying numbers of arguments but does check the data type of the first *n* arguments. If you do not include a value for *n*, **lint** checks no arguments (*n*=0).
- /\*ARGSUSED\*/** Turns on the **-v** flag for the next function.
- /\*LINTLIBRARY\*/** If you place this comment at the beginning of a file, **lint** does not identify unused functions in the file.

The **lint** command first writes messages about each source file as it processes the file. It collects messages about included files and writes those after it has gone through all the source files. Finally, if you have not specified the **-c** flag, it collects information gathered from all input files and checks it for consistency. At this point, if it is not clear whether a message stems from a given source file or from one of its included files, **lint** displays the source file name followed by a question mark.

## Flags

- a** Suppresses messages about assignments of long values to variables that are not long.
- b** Suppresses messages about unreachable break statements.
- h** Does not try to detect bugs, improve style, or reduce waste.
- c** Causes **lint** to produce a **.ln** file for every **.c** file on the command line. These **.ln** files are the product of the first pass of **lint** only and are not checked for inter-function compatibility.
- lkey** Includes the additional **lint** library **llib-lkey.ln**. You can include a **lint** version of the math library **llib-lm.ln** by specifying **-lm** on the command line or **llib-ldos.ln** by specifying **-ldos** on the command line. Use this flag to include local **lint** libraries when checking files that are part of a project having a large number of files. This flag does not prevent **lint** from using the **llib-lc.ln** library.
- n** Does not check for compatibility with either the standard or the portable **lint** libraries.

# lint

---

- Nnnum** Increases the size of the symbol table. The default size is 1500.
- o lib** Causes **lint** to create a lint library with the name **llib-lib.ln**. The **-c** flag nullifies any use of the **-o** flag. The lint library produced is the input that is given to the second pass of **lint**. The **-o** flag simply causes this file to be saved in the named lint library. To produce a **llib-lib.ln** without extraneous messages, use the **-x** flag. The **-v** flag is useful if the source files for the lint library are just external interfaces (for example, the way the file **llib-1c** is written). These flag settings are also available through the use of lint comment lines.
- p** Checks for portability to other C dialects.
- u** Suppresses messages about functions and external variables that are either used and not defined or defined and not used. Use this flag to run **lint** on a subset of files of a larger program.
- v** Suppresses messages about function parameters that are not used.
- x** Suppresses messages about variables that have external declarations but are never used.

In addition, **lint** recognizes the following flags of the **cpp** command (macro preprocessor):

- Dname[=def]** Defines the **name**, as if by **#define**. The default *def* is **1**.
- Idir** Adds *dir* to the list of directories in which **lint** searches for **#include** files.
- Uname** Removes any initial definition of **name**, where **name** is a reserved symbol that is predefined by the particular preprocessor.

## Examples

1. To check a C program for errors:

```
lint program.c
```

2. To suppress some of the messages:

```
lint -v -x program.c
```

This checks `program.c`, but does not display error messages about unused function parameters (**-v**) or unused externals (**-x**).

3. To check the program against an additional lint library:

```
lint -lsubs program.c
```

This checks `program.c` against both the standard lint library (`/usr/lib/llib-1c.ln`) and `/usr/lib/llib-1subs.ln`.

4. To check against the portable library and an additional library:

```
lint -lsubs -p program.c
```

This checks `program.c` against both the portable lint library (`/usr/lib/lilib-port.ln`) and `/usr/lib/lilib-lsubs.ln`.

5. To check against a nonstandard library only:

```
lint -lsubs -n program.c
```

This checks `program.c` against only `/usr/lib/lilib-lsubs.ln`.

## Files

<code>/usr/lib/lint[12]</code>	Programs.
<code>/usr/lib/lilib-lc.ln</code>	Declarations for standard functions (binary format).
<code>/usr/lib/lilib-lc</code>	Declarations for standard functions (source).
<code>/usr/lib/lilib-port.ln</code>	Declarations for portable functions (binary format).
<code>/usr/lib/lilib-port</code>	Declarations for portable functions (source).
<code>/usr/lib/lilib-lm.ln</code>	Declarations for standard math functions (binary format)
<code>/usr/lib/lilib-lm</code>	Declarations for standard math functions (source)
<code>/usr/lib/lilib-ldos.ln</code>	Declarations for DOS Services functions (binary format).
<code>/usr/lib/lilib-ldos</code>	Declarations for DOS Services functions (source).
<code>/usr/tmp/*lint*</code>	Temporary files.

## Related Information

The following command: “`cc`” on page 112.

The topic “Checking C Programs” in *AIX Operating System Programming Tools and Interfaces*.

The “Overview of International Character Support” in *Managing the AIX Operating System*.



# ln

---

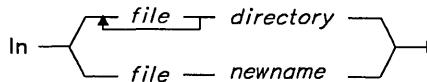
# ln

---

## Purpose

Links files.

## Syntax



OL805028

## Description

The **ln** command links *file* to *newname* (in the current directory), or to the same name (*file*) in another existing *directory*. You can link directories, provided the two directories have the same parent.

If you are linking a file to a new name, you can list only one *file*. If you are linking to a *directory*, you can list more than one *file*.

**Note:** You cannot link files across file systems.

## Examples

1. To create another name (also called an alias) for a file:

```
ln chap1 intro
```

This links `chap1` to the new name `intro`. If `intro` does not already exist, the file name is created. If `intro` does exist, the file is replaced by a link to `chap1`. Now `chap1` and `intro` are two file names that refer to the same file. Any changes made to one also appear in the other. If one name is deleted with **del** or **rm**, the file is not actually deleted, but remains under the other name.

2. To link a file to the same name in another directory:

```
ln index manual
```

This links `index` to the new name `manual/index`.

**Note the difference:** `intro` in Example 1 is the name of a file; `manual` in Example 2 is a directory that already exists.

3. To link several files to names in another directory:

```
ln chap2 jim/chap3 /u/manual
```

This links chap2 to the new name /u/manual/chap2 and jim/chap3 to /u/manual/chap3.

4. To use **ln** with pattern-matching characters:

```
ln manual/*
```

This links all files in the directory manual into the current directory (.), giving them the same names they have in manual. Note that you must type a space between the asterisk and the period.

## Related Information

The following commands: “**rm**” on page 601, “**mv**” on page 502, and “**cp**” on page 156.

The **chmod** and **link** system calls in *AIX Operating System Technical Reference*.

## locator

---

## locator

---

### Purpose

Controls the sample rate of the locator.

### Syntax

```
locator — -rate —
```

OL805444

### Description

The **locator** command sets the *rate* at which the system checks, per second, the cursor position controlled by the mouse. You can specify any of the following *rates*: **10**, **20**, **40**, **60**, **80**, or **100**. Initially, at system startup, this rate is set at **60**.

**Note:** You can run the **locator** command only from the system console.

### Flag

*-rate* Sets the sampling *rate* to the specified value.

### Example

To set the locator rate to **40**:

```
locator -r40
```

---

# login

---

## Purpose

Allows you to sign on to the system.

## Syntax

login —<sup>1</sup>

<sup>1</sup>This command is not normally entered on the command line.

OL805005

## Description

The **login** program logs you onto the system. Its primary functions are:

- To validate your password.
- To make the required accounting and log entries.
- To set up your processing environment.
- To run the command interpreter that is specified in the password file, usually the **sh** program.

A **logger process**, initially running the **getty** program, is started for each enabled port. **getty** reads a login name and sets work station modes (see “**getty**” on page 372). Then it runs **login**, which may ask for a password. If you do not have a password, press the **Enter** key.

Your log in attempt might fail for the following reasons:

- Your login name/password pair does not match an entry in the password file.
- Your password has expired. This can happen if your system requires that you change your password after a set number of days. In this case, **login** runs the **passwd** command instead of letting you log in. (For more information, see “**passwd**” on page 546.) After you change your password, you can attempt to log in again.
- The system has reached the limit of simultaneously logged-in users. Each AIX kernel sets a limit on the number of concurrent log ins by nonprivileged users; this limit may be one. A **privileged** user is one that has a user ID from 0 to 20. A privileged user can log in at any time.

In one special case, **login** does not ask for a user name and password pair. When the login port is the console and the file **/etc/autolog** contains a valid user name, **login** creates a login session for that user automatically. Other processing by **login** proceeds normally.

## login

---

When a user logs in successfully, the **login** program makes entries in **/etc/utmp**, the record of users logged into the system, and in **/usr/adm/wtmp** (if it exists), for use in accounting. On invalid login attempts (due to an incorrect login name or password), **login** makes entries in the **/etc/.ilog** file.

Once you are logged in to the system, the **login** program lists your previous login time and the system message of the day (stored in the **/etc/motd** file), if it has been modified since your last log in or if this is your first log in of the day. If a user file size limit has been specified in the **passwd** file, the limit is set with **ulimit** system call. When you log in as user **root** or **su** and the **/etc/.ilog** file is not empty, you see a message advising the you to check the **/etc/.ilog** file for a record of unsuccessful login attempts.

The **login** program sets the **LOGNAME** and **HOME** environment variables from information in the password file. Environment variables inherited from **getty** and **init** (such as those specified in **/etc/environment**) are kept. You may expand or modify the environment by supplying additional parameters to **login** when it requests your login name. These may take the form *xxx* or *xxx=yyy*. Parameters without an equal sign are placed in the environment as **Lnum=xxx**, where *num* is a number starting at 0 and incremented each time a new variable name is required. Parameters containing an equal sign are placed into the environment without modification. If they already exist, the new assignment replaces the older value. There are two exceptions: You cannot change the shell variables **PATH** and **SHELL**. (This restriction prevents people, logging into restricted environments, from spawning secondary shells that are not restricted.) Both **login** and **getty** understand simple single-character quoting conventions. Typing a backslash (\) in front of a character quotes it and allows you to include such things as spaces and tab characters.

The **login** command changes the current directory to your **HOME** directory, changes the ownership of the **port** (work station) to the user logging in, sets the user-and group-IDs of the process, and then runs the program specified for the user in the **passwd** file, normally the shell (**/bin/sh**). **login** calls this program with a name consisting of - (minus) followed by the last segment of its path name. An instance of the shell can therefore determine from its invocation name whether it is a login shell or a subshell.

The **/etc/passwd** file entry may include parameters that are always passed to the shell program. For more details, see the **passwd** file in *AIX Operating System Technical Reference*.

## Files

<b>/etc/utmp</b>	Accounting.
<b>/usr/adm/wtmp</b>	Accounting.
<b>/etc/.ilog</b>	Accounting.
<b>/etc/autolog</b>	Login ID for automatic login.
<b>/usr/motd</b>	Message of the day.
<b>/etc/passwd</b>	Password file.
<b>.llog</b>	Date of last login.

## Related Information

The following commands: “**users**” on page 802, “**getty**” on page 372, “**init**” on page 396, “**passwd**” on page 546, and “**penable**” on page 550.

The **passwd** and **utmp** files in *AIX Operating System Technical Reference*.

The discussion of login sessions in *Managing the AIX Operating System*.

# logname

---

## logname

---

### Purpose

Displays your login name.

### Syntax

logname —

OL805145

### Description

The **logname** command writes to standard output the name you used to log into the system. It is the contents of the environment variable **\$LOGNAME**, which is set when you log into the system.

### Files

*/etc/profile*

### Related Information

The following commands: “**env**” on page 298 and “**login**” on page 453.

The **logname** subroutine *AIX Operating System Technical Reference*.

The **environ** special facility in *AIX Operating System Technical Reference*.

---

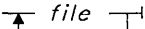
# lorder

---

## Purpose

Finds the best order for member files in an object library.

## Syntax

lorder 

OL805029

## Description

The **lorder** command reads one or more object or library archive *files*, looking for external references and writing a list of paired file names to standard output. The first of each paired files contains references to identifiers that are defined in the second file. You can send this list to the **tsort** command to find an ordering of a library member file suitable for one-pass access by **ld**.

If object files do not end with **.o**, **lorder** overlooks them and attributes their global symbols and references to some other file.

## Example

To create a subroutine library:

```
lorder charin.o scanfld.o scan.o scanln.o | tsort | xargs ar qv libsubs.a
```

This creates a subroutine library named `libsubs.a` that contains `charin.o`, `scanfld.o`, `scan.o`, and `scanln.o`. The ordering of the object modules in the library is important. The **ld** command requires each module to precede all the other modules that it calls or references. The **lorder** and **tsort** commands together add the subroutines to the library in the proper order.



## lorder

---

Suppose that `scan.o` calls `scanfld.o` and `scanln.o`. `scanfld.o` also calls `charin.o`. First, the **lorder** command creates a list of pairs that shows these dependencies:

```
charin.o charin.o
scanfld.o scanfld.o
scan.o scan.o
scanln.o scanln.o
scanfld.o charin.o
scanln.o charin.o
scan.o scanfld.o
```

Next, the `|` (vertical bar) sends this list to the **tsort** command, which converts it into the ordering we need:

```
scan.o
scanfld.o
scanln.o
charin.o
```

Note that each module precedes the module it calls. `charin.o`, which does not call another module, is last.

The second `|` then sends this list to **xargs**, which constructs and runs the following **ar** command:

```
ar qv libsubs.a scan.o scanfld.o scanln.o charin.o
```

This **ar** command creates the properly ordered library.

## Files

`/tmp/sym*` Temporary files

## Related Information

The following commands: “**ar**” on page 58, “**ld**” on page 427, “**nm**” on page 521 “**tsort**” on page 778, and “**xargs**” on page 857.

The **ar** file in *AIX Operating System Technical Reference*.

---

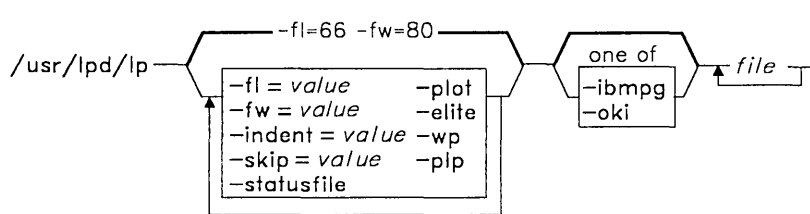
# lp

---

## Purpose

Prints a file in a format suitable for sending to a line printer.

## Syntax



OL805396

## Description

The **lp** command prints *file* on its standard output in a form that is suitable for a line printer. The **lp** command is normally invoked by the **qdaemon** command. **qdaemon** directs the output from **lp** to the appropriate device.

Flags are passed to **lp** in the following ways:

- Flags specified in the **qconfig** structure are passed each time that **lp** is invoked. The **-plp**, **-ibmpg**, **-oki**, and **-statusfile** flags most likely appear in **qconfig**.
- Flags that are not recognized by the **print** command are assumed to be for **lp** and are passed to **lp** with the requested job.

## Flags

- elite** Prints the text at 12 characters per inch instead of 10 characters per inch. This flag changes the default forms width to 96 characters.
- fl = value** Sets the forms length equal to *value*. The default length is 66 lines.
- fw = value** Sets the forms width equal to *value*. The default width is 80 columns. Lines that are wider than *value* are truncated. If you set *value* to 0, no truncation is performed.
- ibmpg** Specifies an IBM Graphic Printer.

- indent = value**    Indents the printed output the number of spaces specified with *value*.
- oki**                Specifies an Okidata Model 92 or 93.
- plot**              Passes text directly to the printer without processing. This is useful when using the printer as a plotter. Normally, lines that contain backspaces and carriages return characters are processed so that they print with minimum print head motion. The sequence ESC-9 maps to half line feeds.
- plp**                Sets and resets printer port parameters, if the printer is attached with a parallel interface.
- skip = value**      Does not print the first *value* blank lines in the file.
- statusfile**        Updates the status information in the status file that is open on file descriptor 3. The status information is passed from **qdaemon**.
- wp**                 Sets the printer, if possible, to the Word Processing mode.

## Related Information

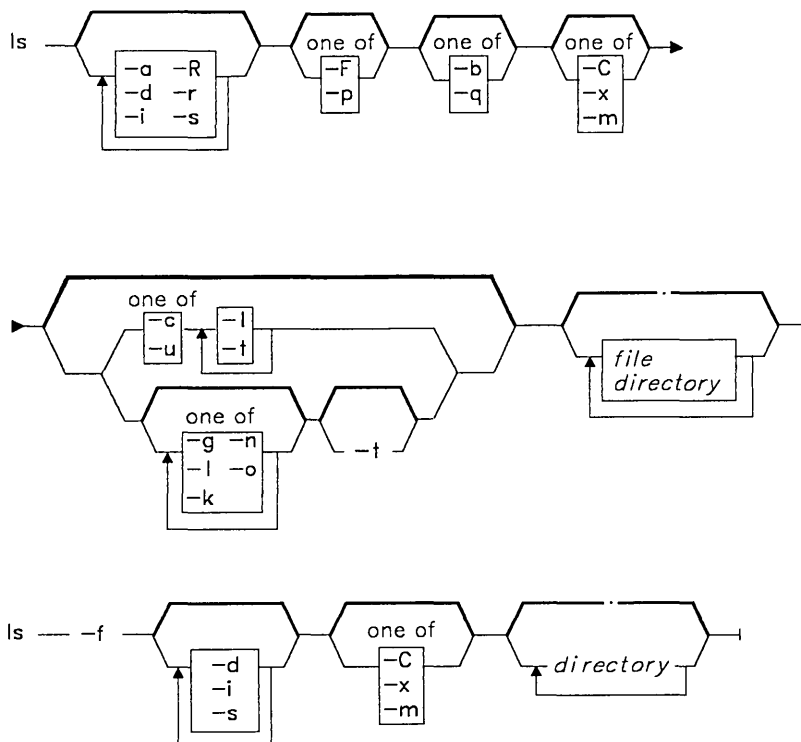
The following commands: “**print**” on page 566 and “**qdaemon**” on page 590.  
The **qconfig** file in *AIX Operating System Technical Reference*.

## ls

## Purpose

Displays the contents of a directory.

## Syntax



OL805030

OL805243

## Description

The `ls` command writes to standard output the contents of each specified *directory* or the name of each specified *file*, along with any other information you ask for with the flags. If you do not specify a *file* or *directory*, `ls` displays the contents of the current directory. By default, `ls` displays all information in alphabetic order by file name. The collating

**ls**

sequence is determined by the **NLCTAB** environment variable (see “**ctab**” on page 204). Individual file names are listed before directory names.

There are three main ways to format the output:

1. List one entry per line. This is the default format.
2. List entries in multiple columns by specifying either the **-C** or **-x** flags.
3. List entries in a comma-separated series by specifying the **-m** flag.

To determine the number of character positions in the output line, **ls** uses the environment variable **COLUMNS**. If this variable is not set, it reads the **terminfo** file. If **ls** cannot determine the number of character positions by either of these methods, it uses a default value of 80.

The mode displayed with the **-l** flag is interpreted as follows:

If the first character is:

- d** The entry is a directory.
- b** The entry is a block special file.
- c** The entry is a character special file.
- p** The entry is a first-in first-out (FIFO) special file.
- The entry is an ordinary file.
- D** The entry is a remote directory.
- F** The entry is a remote ordinary file.
- B** The entry is a remote block special file.
- C** The entry is a remote character special file.
- P** The entry is a remote first-in first-out (FIFO) special file.

The next nine characters are divided into three sets of three characters each. The first three characters show the owner’s permission. The next set of three characters show the permission of the other users in the group. The last set of three characters show the permission of any one else with access to the file. The three characters in each set show read, write and execute permission of the file. Execute permission of a directory lets you search a directory for a specified file.

Permissions are indicated as follows:

- r** You can read the file.
- w** You can edit (write) the file.
- x** You can search the file.
- You do not have permission to the file.

The group-execute permission character is **s** if the file has set-group ID mode. The user-execute permission character is **S** if the file has set-user-ID mode. The last character of the mode (normally **x** or **-**) is **t** if the 1000 (octal) bit of the mode is set; see “**chmod**” on page 128 for the meaning of this mode. The indications of set-ID and 1000 bit of the mode are capitalized (**S** and **T** respectively) if the corresponding execute permission is not set.

When the size of the files in a directory are listed, the **ls** command displays a total count of blocks, including indirect blocks.

The environment variables **NLLDATE** and **NLTIME** control the format of the date and time. The environment variable **NLSMONTH** controls the short names of months.

## Flags

- a** Lists all entries in the directory including the entries that begin with a . (dot).
- b** Displays nonprintable characters in an octal `\nnn` notation.
- c** Uses the time of last modification of the i-node (file created, mode changed, and so on) for sorting (when used with **-t**) or for displaying (when used with **-l**). This flag has no effect when not used with either **-t** or **-l** or both.
- C** Sorts output vertically in a multi-column format.
- d** Displays only the information for the directory named. This is useful with the **-l** flag to get the status of a directory.
- f** Lists the name in each slot for each named *directory*. This flag turns off **-l**, **-t**, **-s**, and **-r**, and turns on **-a**; the order is the order in which entries appear in the directory.
- F** Puts a / (slash) after each file name if the file is a directory and an \* (asterisk) after each file name if the file can be executed.
- g** Displays the same information as with **-l**, except for the owner.
- i** Displays the i-number in the first column of the report for each file.
- k** Displays the permission codes, node ID, remote UID, remote GID, time of last modification, size (in bytes), and file name for remote entries.  
  
For remote files and directories, the local owner and local group are obtained by using inverse IDs. If there is no inverse ID or if **ls** cannot determine the inverse ID, a - (minus sign) displays in the corresponding field. If possible, remote nodes are identified with nicknames. Otherwise, they are identified by their NID displayed in hexadecimal. (See “Distributed Services Concepts” in *Managing the AIX Operating System*.)  
  
For local files and directories that do not have a nickname defined for the local node ID, the node ID field displays as a - (minus sign), and the raw UID (GID) field contains the local owner UID (group GID).
- l** Displays the mode, number of links, owner, group, size (in bytes), and time of last modification for each file. If the file is a special file, the size field will instead contain the major and minor device numbers.
- m** Uses stream output format (a comma-separated series).
- n** Displays the same information as with **-l**, except that it displays the user and the group IDs instead of the user and group names.
- o** Displays the same information as with **-l**, except for the group.

- p Puts a slash after each file name if that file is a directory. This is useful when you pipe the output of `ls` to the `pr` command as follows:  

```
ls -p | pr -5 -t -w80
```
- q Displays nonprintable characters in file names as the character `?`.
- r Reverses the order of the sort, giving reverse alphabetic or the oldest first, as appropriate.
- R Lists all subdirectories recursively.
- s Gives size in blocks (including indirect blocks) for each entry.
- t Sorts by time of last modification (latest first) instead of by name.
- u Uses the time of the last access instead of time of the last modification for sorting (when used with `-t`) or for displaying (when used with `-l`). This flag has no effect when not used with either `-t` or `-l` or both.
- x Sorts output horizontally in a multi-column format.

## Examples

1. To list all files in the current directory:

```
ls -a
```

This lists all files, including `.` (dot), `..` (dot-dot), and other files with names beginning with a dot.

2. To display detailed information:

```
ls -l chap1 .profile
```

This displays a long listing with detailed information about `chap1` and `.profile`.

3. To display detailed information about a directory:

```
ls -d -l . manual manual/chap1
```

This displays a long listing for the directories `.` and `manual`, and for the file `manual/chap1`. Without the `-d` flag, this would list the files in `.` and `manual` instead of the detailed information about the directories themselves.

4. To list the files in order of modification time:

```
ls -l -t
```

This displays a long listing of the files that were modified most recently, followed by the older files.

## Files

/etc/passwd	Contains user IDs.
/etc/group	Contains group IDs.
/usr/lib/terminfo/*	Contains terminal information.

## Related Information

The following commands: “**chmod**” on page 128, “**ctab**” on page 204, and “**find**” on page 326.

The **environment** miscellaneous facility in *AIX Operating System Technical Reference*.

“Overview of International Character Support” in *Managing the AIX Operating System*.

“Distributed Services Concepts” in *Managing the AIX Operating System*.



ls

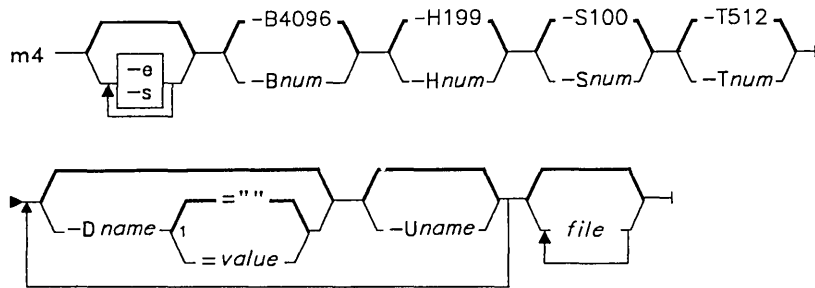
---

# m4

## Purpose

Preprocesses files, expanding macro definitions.

## Syntax



OL805434

## Description

The **m4** command is a macro processor used as a preprocessor for C and other languages. You can use it to process built-in macros or user-defined macros. Each *file* is processed in order. If you do not specify a *file* or if you give a minus (-) as a file name, **m4** reads standard input. It writes the processed macros to standard output.

Macro calls follow the form:

```
macroname(argument . . . )
```

The left parenthesis must immediately follow *macroname*. If the left parenthesis does not follow the name of a defined macro, **m4** reads it as a macro call with no arguments. Macro names consist of ASCII alphabetic letters, digits, and the underscore character (`_`). Extended characters are not allowed in macro names. The first character cannot be a digit.

While collecting arguments, **m4** ignores unquoted leading blanks, tabs, and new-line characters. Use single quotation marks to quote strings. The value of a quoted string is the string with the quotation marks stripped off.

When **m4** recognizes a macro, it collects arguments by searching for a matching right parenthesis. If you supply fewer arguments than appear in the macro definition, **m4** considers the trailing arguments in the definition to be null. Macro evaluation proceeds

normally during the collection of the arguments. All commas or right parentheses within the value of a nested call are translated literally; they do not need an escape character or quotation marks. After collecting arguments, **m4** pushes the value of the macro back onto the input stream and scans again.

## Built-in Macros

The **m4** command makes available the following built-in macros. You may redefine them, but you will lose the original meaning. The values of these macros are null unless otherwise stated:

- define**(*name,new\_name*) Replaces the macro *name* with the value of *new\_name*. The *new\_name* string can take the form  $\$n \dots$  (where *n* is a digit). In this case, each occurrence of *n* in the replacement text is replaced by the *n*-th argument of *name*.  $\$0$  is the name of the macro. The null string replaced missing arguments. The number of arguments replaces  $\#\$ . A comma-separated list of all arguments replaces  $\*\$ .  $\$@$  acts like  $\*\$ , but each argument is quoted with the current quotation character (see **changequote**).
- undefine**(*name*) Removes the definition of *name*.
- defn**(*name . . .*) Returns the quoted definition of *name*.
- pushdef**(*name,new\_name*) Redefines *name* with *new\_name* as in *define*, but save any previous definition.
- popdef**(*name . . .*) Removes the current definition of *name* and returns to the previous definition, if one existed.
- ifdef**(*name,true,[false]*) Returns the value of *true* only if *name* is defined, otherwise return *false*. If you do not supply *false*, its value is null.
- Note:** The word **unix** is predefined.
- shift**(*argument . . .*) Returns all but the first argument. The other arguments are quoted and pushed back with commas in between. The quoting nullifies the effect of the extra scan that will subsequently be performed.
- changequote**(*L,R*) Changes quote symbols to *L* and *R*. The symbols can be up to five bytes long. **changequote** without arguments restores the original values ( ` ' ).
- changecom**(*Lcom,Rcom*) Changes left and right comment markers from the default  $\#$  and new-line character to *Lcom* and *Rcom*. With no arguments, the comment mechanism is disabled. With one argument, the left marker becomes the parameter and the right marker becomes a new-line character. With two arguments, both

---

	markers are affected. Comment markers can be up to five bytes long.
<b>divert</b> ( <i>num</i> )	Changes the current output stream to stream <i>num</i> . There are 10 output streams, numbered 0-9. The final output is the concatenation of the streams in numerical order. Initially, stream 0 is the current stream. <b>m4</b> discards output diverted to a stream other than 0-9.
<b>undivert</b> ( <i>num</i> . . . )	Causes immediate output of text from the specified diversions (or all diversions if there is no argument). Text may be undiverted into another diversion. Undiverting discards the diverted text.
<b>divnum</b>	Returns the value of the current output stream.
<b>dnl</b>	Reads and discards characters up to and including the next new-line character.
<b>ifelse</b> ([ <i>string1</i> , <i>string2</i> , <i>true</i> ,[ <i>false</i> ]] . . . )	If <i>string1</i> and <i>string2</i> are the same, then the value is <i>true</i> . If they are not and if there are more than four arguments, <b>m4</b> repeats the process with the additional arguments (4, 5, 6, and 7). Otherwise, the value is either <i>false</i> or null if you provide no value for <i>false</i> .
<b>incr</b> ( <i>num</i> )	Returns the value of its argument incremented by 1.
<b>decr</b> ( <i>num</i> )	Returns the value of its argument decreased by 1.
<b>eval</b> ( <i>expr</i> [, <i>num1</i> [, <i>num2</i> ]])	Evaluates its first argument as an arithmetic expression, using 32-bit arithmetic. The operators you can use include +, -, *, /, %, ^ (exponentiation), bitwise &,  , ~, and ^ relationals, and parentheses. Octal and hex numbers can be specified as in C. <i>num1</i> specifies the radix for the result of the expression. The default radix is 10. The optional <i>num2</i> specifies the minimum number of digits in the result.
<b>len</b> ( <i>string</i> )	Returns the number of bytes in <i>string</i> .
<b>dlen</b> ( <i>string</i> )	Returns the number of displayable characters in <i>string</i> ; that is, two-byte extended characters are counted as one displayable character.
<b>index</b> ( <i>s1</i> , <i>s2</i> )	Returns the position in the string <i>s1</i> where the string <i>s2</i> begins (zero origin), or -1 if the second parameter does not occur.
<b>substr</b> ( <i>string</i> , <i>position</i> , <i>num</i> )	Returns a substring of <i>string</i> . The beginning of the substring is selected with <i>position</i> , and <i>num</i> indicates the length of the substring. Without <i>num</i> , the substring includes everything to the end of the first string.

<b>translit</b> ( <i>string,from,to</i> )	Transliterates the characters in <i>string</i> from the set given by <i>from</i> to the set given by <i>to</i> . No abbreviations are permitted. Two-byte extended characters are correctly mapped into the corresponding replacement characters.
<b>include</b> ( <i>file</i> )	Returns the contents of <i>file</i> or displays an error message if it cannot access the file.
<b>sinclude</b> ( <i>file</i> )	Returns the contents of <i>file</i> , but it gives no error message if <i>file</i> is inaccessible.
<b>syscmd</b> ( <i>command</i> )	Runs the AIX <i>command</i> . No value is returned.
<b>sysval</b>	Returns the return code from the last call to <i>syscmd</i> .
<b>maketemp</b> ( . . . <b>XXXXXX</b> . . . )	Replaces <b>XXXXXX</b> in its argument with the current process ID number.
<b>m4exit</b> ( <i>value</i> )	Exits from <b>m4</b> immediately, returning the specified exit <i>value</i> (the default is 0).
<b>m4wrap</b> ( <i>lastmacro</i> )	Runs <i>lastmacro</i> after reading the end-of-file character. For example: <code>m4wrap('cleanup()')</code> runs the <code>cleanup</code> macro at the end of <b>m4</b> .
<b>errprint</b> ( <i>message</i> )	Includes <i>message</i> on the diagnostic output file.
<b>dumpdef</b> ([ <i>name</i> . . . ])	Writes to standard output the current names and definitions for the named items or for all if no arguments are provided.
<b>traceon</b> ( <i>macro</i> )	Turns on tracing for <i>macro</i> . If none is named, tracing is turned on for all macros.
<b>traceoff</b> ( <i>macro</i> . . . )	Turns off trace globally and for any <i>macro</i> specified. Macros specifically traced by <b>traceon</b> can be untraced only by specific calls to <b>traceoff</b> .

## Flags

<b>-Bnum</b>	Makes <i>num</i> the size of the push-back and parameter collection buffers (the default is 4096).
<b>-e</b>	Operates interactively. Interrupts are ignored and the output is not buffered.
<b>-Hnum</b>	Makes <i>num</i> the size of the symbol table hash array (the default is 199). The size must be a prime number.
<b>-s</b>	Enables the line sync output for the C preprocessor ( <code>#line . . .</code> ).

- Snum**            Makes *num* the size of the call stack (the default is 800 slots). Macros take three slots, and nonmacro arguments take one.
- Tnum**            Makes *num* the size of the token buffer (the default is 512 bytes).
- The preceding flags must appear before any file names and before any **-D** or **-U** flags.
- Dname[=val]**    Define *name* as *val*. If *val* is not specified, *name* becomes null.
- Uname**            Undefines a *name* previously defined with the **-D** flag.

## Example

To preprocess a C language program with **m4** and compile it:

```
m4 prog.m4 >prog.c
cc prog.c
```

## Related Information

The following commands: “**cc**” on page 112 and “**cpp**” on page 163.

The “Overview of International Character Support” in *Managing the AIX Operating System*.

# mail

---

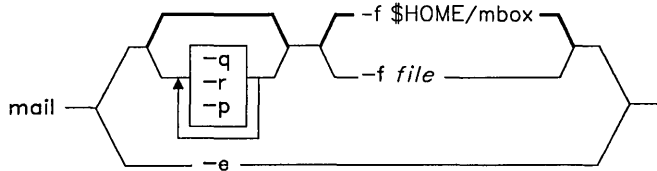
## mail

---

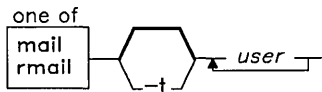
### Purpose

Sends messages to system users and displays messages from system users.

### Syntax



OL805347



OL805034

### Description

The **mail** command with no flags writes to standard output, one message at a time, all stored mail addressed to the your login name. Following each message, **mail** prompts you with a question mark (?). Press the **Enter** key to display the next mail message, or enter one of the subcommands which control the disposition of the message (see “Subcommands” on page 471).

When sending mail, you specify *users*, and then **mail** reads a message from standard input until you press END OF FILE (**Ctrl-D**) or enter a line containing only a period (.). It prefixes this message with the sender’s name and the date and time of the message (its *postmark*) and adds this message to the file `/usr/mail/user` for each *user* specified on the command line.

The action of **mail** can be modified in two ways by manipulating `/usr/mail/user`:

1. The default permission assignment for “others” is “read-only.” If you change this permission assignment to “read/write” or to all permissions denied, the system preserves the file, even when it is empty, in order to maintain the desired permissions.

2. You can edit the file to contain as its first line:

Forward to *person*

This causes all messages sent to *user* to be sent to *person* instead. The **Forward to** feature is especially useful for sending all of a person's mail to a particular machine in a network environment.

The **rmail** command is similar to **mail** except that it only allows the user to send mail. The **uucp** command uses **rmail** for security reasons, allowing a user to send mail to a remote system, but not to read mail on that system.

To specify a recipient on a remote system, prefix the system name and an exclamation mark (!) to *user*. See "**uucp**" on page 807 for a detailed discussion of how to address remote systems.

## Flags

- e** Does not display any messages. This flag causes **mail** to return an exit value of 0 if the user has mail, an exit value of 1 if he has no mail.
- f file** Saves mail in the named *file* instead of in the default mailfile, **\$HOME/mbox**.
- p** Displays mail without prompting for a disposition code. This flag does not delete, copy, or forward any messages. (For disposition codes, see "Subcommands").
- q** Causes **mail** to exit when you press INTERRUPT (**Alt-Pause**). Normally, pressing INTERRUPT (**Alt-Pause**) stops only the message being displayed. (In this case, the next message sometimes does not display until you enter the **p** subcommand.)
- r** Displays mail in first-in, first-out order.
- t** Prefixes each message with the names of all recipients of the mail. (Normally, only the individual recipient's name appears as addressee.)

Usually, *user* is a name recognized by the **login** command. It can also be the ASCII synonym that is automatically defined for any name that contains NLS code points. If the system does not recognize one or more of the specified *users* or if **mail** is interrupted during input, **mail** saves messages in the file **\$HOME/dead.letter** to allow for editing and resending.

## Subcommands

The following subcommands control message disposition:

- +** Displays the next mail message (the same as pressing the **Enter** key).
- Displays the previous message.



## mail

---

<b>d</b>	Deletes the current message and displays the next message.
<b>p</b>	Displays the current message again.
<b>s</b> <i>[file]</i>	Saves the message in the named <i>file</i> instead of in the default mailfile, <b>\$HOME/mbox</b> .
<b>w</b> <i>[file]</i>	Saves the message, without its postmark, in the named <i>file</i> instead of in the default mailfile, <b>\$HOME/mbox</b> .
<b>m</b> <i>user</i>	Forwards the message to the named <i>user</i> .
<b>q</b>	Writes any mail not yet deleted to <b>/usr/mail/user</b> and exits. Pressing END OF FILE ( <b>Ctrl-D</b> ) has the same effect.
<b>x</b>	Writes all mail unchanged to <b>/usr/mail/user</b> and exits.
<b>!AIX-cmd</b>	Runs the specified AIX command.
<b>*</b>	Displays a subcommand summary.

## Examples

1. To display your mail:

```
mail
```

After the most recent message is displayed, a ? (question mark) indicates that **mail** is waiting for one of the subcommands explained previously (+, -, **d**, **p**, etc.). Enter \* (asterisk) to list the subcommands available.

2. To send mail to other users:

```
mail tom rachel  
Don't forget the  
meeting tomorrow at 9:30.
```

### **Ctrl-D**

This mails the message Don't forget the meeting tomorrow at 9:30. to users tom and rachel. The **Ctrl-D** indicates the end of the message, but is not included in the text.

3. To send a file to another user:

```
mail fran <proposal
```

This sends the contents of the file `proposal` to `fran`. You can create `memo` with an editor, which allows you to correct your mistakes before sending the message. You can also use this form of the **mail** command to send someone a copy of a data file.

4. To retrieve a file that was sent to you:

```
mail
```

This displays the messages mailed to you one at a time. You need to look at them because the file you want was actually added to `/usr/mail/user` as a message. You may see several other messages before the file that was sent to you. If so, press the **Enter** key after the `?` prompt until the desired file appears. If you go too far, enter the `-` (minus) subcommand to go back a message. After the `?` immediately following the file, enter:

```
w mycopy
```

This creates a file named `mycopy` in the current directory that contains the text mailed to you. Actually, you can save a copy of any message this way.

## Files

<code>/etc/passwd</code>	To identify sender and locate <i>user</i> .
<code>/usr/mail/user</code>	Incoming mail for <i>user</i> .
<code>\$HOME/mbox</code>	Saved mail.
<code>\$HOME/dead.letter</code>	Unmailable text.
<code>/tmp/ma*</code>	Temporary file.
<code>/usr/mail/*.lock</code>	Lock for mail directory.

## Related Information

The following commands: “**login**” on page 453, “**uucp**” on page 807, and “**write**” on page 853.

# make

---

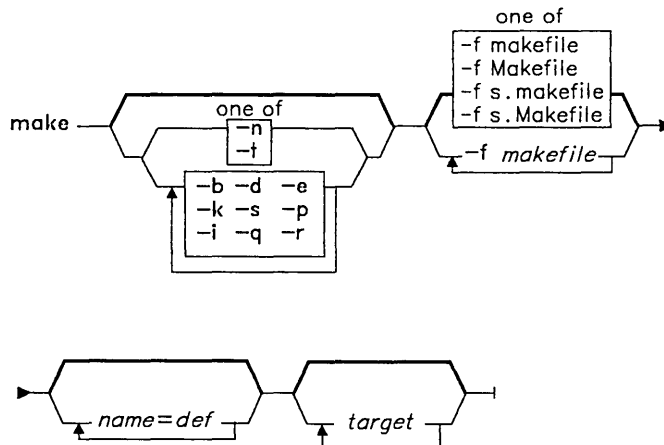
## make

---

### Purpose

Maintains up-to-date versions of programs.

### Syntax



OL805035

### Description

The **make** command reads *makefile* for information about the specified *target* files and for the commands necessary to update them. **make** does not change the *target* if you have not changed any of the source files since you last built it. It considers a missing file to be a changed file (out of date).

You can also include macro definitions on the command line after all of the flags. Macro definitions have the form:

*macro-name* = *string*

See “Macros” on page 477 for more information about macros and their uses.

The **make** command considers all entries on the command line that follow the flags and that do not contain an equal sign to be target file names.

## Description File

The description file contains a sequence of entries specifying the files that the target files depend on. The general form of an entry is:

```
targ [targ] . . . :[:][file] . . . [; cmd] . . . [#]
      [cmd] . . . [#]
```

The first line of an entry (called the *dependency line*), contains a list of targets followed by a : (colon) and an optional list of prerequisite files or dependencies. If you put shell commands on the dependency line, they must be preceded by a ; (semicolon). All commands that follow the semicolon and all following lines that begin with a tab contain shell commands that **make** uses to build the target.

To specify more than one set of commands, you must enter more than one dependency definition. In this case, each definition must have the target name followed by two colons (::), a dependency list, and a command list.

The first line that does not begin with a tab or # (hash sign) begins a new dependency or a macro definition. Command lines are performed one at time, each by its own subshell. Thus, the effect of some shell commands, such as **cd**, does not extend across new-line characters. You can, however, put a \ (backslash) at the end of a line to continue it on the next physical line. A comment begins with a # and ends with a new-line character.

The first one or two characters in a command can be one of the following special characters:

- Ignores errors returned by the command on this line.
- @: Does not display this command line.
- @
- @- Does not display this command line and ignores errors.

## Suffixes

The **make** command has default rules that govern the building of most standard files. These rules depend on the standard suffixes used by the system utility programs to identify file types. These rules define the starting and ending file types so that, for example, given a specified **.o** file, **make** can infer the existence of a corresponding **.c** file and knows to compile it using the **CC -c** command.

Use a ~ (tilde) in the suffix to indicate a SCCS file. For example, the **.C~.o** rule governs changing an SCCS C source file into an object file.

You can define rules within the description file. **make** recognizes as a rule any target that contains no slashes and starts with a dot.

A rule with only one suffix (that is, **.c:**) defines the building of *prog* from all its source files.

## make

---

You can also add suffixes to the list of suffixes recognized by **make** and add to the default dependency rules. Use the target name **.SUFFIXES** followed by the suffixes you want to add. Be careful of the order in which you list the suffixes. **make** uses the first possible name for which both a file and a rule exist. The default list is:

```
.SUFFIXES: .o .c .c~ .y .y~ .l .l~ .s .s~ .sh .sh~ .h .h~
```

You can clear the list of suffixes by including **.SUFFIXES:** with no following list.

### Special Target Names

You can use some special target names in the description file to tell **make** to process the file in a different manner. The special target names are:

- .DEFAULT**      The commands that appear after this name in the description file tell **make** what to do if it can find no commands or default rules to tell it how to create a specific file.
- .IGNORE**        If this name appears on a line by itself, **make** does not stop when errors occur. Using a - (minus) as the first character on a line in the description file tells **make** to ignore errors for the command on that line.
- .PRECIOUS**      The files named on the same line as this special name are not removed when **make** is interrupted.
- .SILENT**        If this name appears on a line by itself, **make** does not display any of the commands that it performs to build a file.
- .SUFFIXES**      Use this name to add more suffixes to the list of file suffixes that **make** recognizes.

### Environment

When you run **make**, it reads the environment and treats all variables as macro definitions. **make** processes the environment variables after it processes its own internal rules and before processing any description files. Therefore, macro assignments in a description file normally override duplicate environment variables. The **-e** flag instructs **make** to use the environment variables instead of the description file macro assignments.

**Note:** Because **make** uses the dollar sign symbol (\$) to designate a macro, do not use that symbol in file names of targets and parents, or in commands in the description file unless you are using a defined **make** macro.

The **make** command recognizes a macro **MAKEFLAGS**, which can be assigned any **make** command line flag except **-f**, **-p**, and **-d**. When **make** begins, it assigns the current flags to **MAKEFLAGS**. It passes this variable to any commands it invokes, including additional invocations of **make** itself. Thus you can perform a **make -n** recursively on a software system to see what would have been performed. The **-n** is put in **MAKEFLAGS** and passed to further copies of the shell that runs the next level of **make** commands. In this way, you can check all of the description files for a software project without actually compiling the project.

## Macros

Entries of the form *string1* = *string2* are macro definitions. *string2* can consist of all characters that can occur on a line before a comment character (#) or before a new-line character that is not a continuation line. After this macro definition, **make** replaces each \$(*string1*) in the file with *string2*. You do not have to use the parentheses around the macro name if the macro name is only one character long and there is no substitute sequence (see the next paragraph).

If you use the following form, you can also replace characters in the macro string with other characters for one time that you use the macro:

```
$(string1[:subst1=[subst2]])
```

The optional *:subst1* = *subst2* + specifies a substitute sequence. If you specify a substitute sequence, **make** replaces each *subst1* in the named macro with *subst2* (if *subst1* does not overlap with another *subst1*). Strings in a substitute sequence begin and end with any of the following: a blank, tab, new-line character, or beginning of line. See “Libraries” on page 478 for an example of the use of the substitute sequence.

## Internal Macros

The **make** command has five internal macros. It assigns values to these macros under one or more of the following conditions:

- When it uses an internal rule to build a file.
- When it uses a **.DEFAULT** rule to build a file.
- When it uses rules in the description file to build a file.
- When the file is a library member.

They are defined as follows:

\$\*     The file name (without the suffix) of the source file.

\$@     The full target name of the current target.

\$<     The source files of an out-of-date module. **make** evaluates this macro when applying inference rules or the **.DEFAULT** rule. For example:

```
.c.o:
  cc -c $<
```

Here, \$< is the equivalent of \$\* and refers to the **.c** file of any out-of-date **.o** file.

\$?     The list of out-of-date files. **make** evaluates this macro when it evaluates explicit rules from *makefile*.

\$\$     The name of an archive library member. **make** evaluates this macro only if the target is an archive library member of the form *lib(file.o)*. In this case, \$@ evaluates to *lib* and \$\$ evaluates to the library member, *file.o*.

## make

---

You can add an uppercase **D** or **F** to indicate “directory part” or “file part,” respectively, to all internal macros except for `$?`. Thus, `$(@D)` refers to the directory part of the name `$@`. If there is no directory part, **make** uses `./`.

### Libraries

If a target name contains parentheses, **make** considers it an archive library. The string within parentheses refers to a library member. Thus, `lib(file.o)` and `$(LIB)(file.o)` both see an archive library which contains `file.o`. (You must have defined the **LIB** macro already.) The expression `$(LIB)+ (file1.o file2.o)` is not legal.

Rules that apply to archive libraries have the form `x.a`, where `.x` is the suffix of the file you want to add to an archive library. For example, `.C.a` indicates a rule that changes any C source file to a library file member. The following lines give the default rule for this change:

```
lib: lib(file.o) lib(file.o) lib(file.o)
    @echo lib is now up to date
.c.a:
    $(CC) -c $(CFLAGS) $<
    ar rv $@ $*.o
    rm -f $*.o
```

`.x` must be different from the suffix of the archive member. Therefore, you cannot have `lib(file.o)` depend upon `file.o`.

Another, but more limited, example of an archive library maintenance rule follows:

```
lib: lib(file.o) lib(file.o) lib(file.o)
    $(CC) -c $(CFLAGS) $(?:.o=.c)
    ar rv lib $?
    rm $? @echo lib is now up to date
.c.a;;
```

This example rule uses a substitute sequence `(.o=.c)` to replace with `.c` files all `.o` files generated by the `$?` macro. The `$?` list is the set of object file names (inside `lib`) with C source files that are out of date. The macro substitution translates `.o` to `.c`.

If this rule appears in your description file, it disables the default `.c.a:` rule, which creates each object file one by one. This type of organization speeds up archive library maintenance, but becomes hard to use if the archive library contains a mix of assembly programs and C programs.

---

## Flags

- b** Recognizes *makefiles* that were written for old versions of **make**.
- d** Displays detailed information about the files and times that **make** examines (debug mode).
- e** Uses environment variables in place of any assignments made within description files. These assignments normally replace environment variables.
- f *makefile*** Reads *makefile* for a description of how to build the target file. If you give only a - (minus) for *makefile*, **make** reads standard input. If you do not use the **-f** flag, **make** looks in the current directory for a description file named **makefile**, **Makefile**, **s.makefile**, or **s.Makefile**. You can specify more than one description file by entering the **-f** flag more than once (with its associated *makefile* parameter).
- i** Ignores error codes returned by commands. **make** normally stops if a command returns a nonzero code. Use this flag to compile several modules only if you want **make** to continue when an error occurs in one of the modules. Do not link the resulting modules when you use this flag.
- k** Stops processing the current target if an error occurs, but continues with other branches that do not depend on that target.
- n** Displays commands, but do not run them. Displays lines beginning with an @ (at sign). If the command in the description file contains the string `$(MAKE)`, perform another call to **make** (see the discussion of the **MAKEFLAGS** macro on page 476). Use this flag to preview the performance of **make**.
- p** Displays the complete set of macro definitions and target descriptions before performing any commands.
- q** Returns a zero status code if the target file is up to date; returns a nonzero status code if the target file is not up to date.
- r** Does not use the default rules.
- s** Does not display commands on the screen as they are performed.
- t** Changes only the date of the files, rather than performing the listed commands. Use this flag if you have made only minor changes to a source file that do not affect anything outside of that file. This flag changes the date of all target files that appear on the command line or in the description file.



## make

---

### Examples

1. To make the file specified by the first entry in the description file:

```
make
```

2. To display, but not run, the commands that **make** would use to make a file:

```
make -n search.o
```

You may want to do this to verify that a new description file is correct before using it.

3. To save the internal rules in a file:

```
make -p -f /dev/null 2> /dev/null > defaults
```

This lists the internal rules and macros and saves them in the file `defaults` for viewing or editing. All exported shell environment variables are included in the list of macro definitions.

### Files

```
Makefile  
makefile  
s.Makefile  
s.makefile
```

### Related Information

The discussion of **make** in *AIX Operating System Programming Tools and Interfaces*.

# makekey

---

## Purpose

Generates an encryption key.

## Syntax

```
/usr/lib/makekey —
```

OL805240

## Description

The **makekey** command generates an encryption *key* to use with programs that perform encryption. Its input and output are usually pipes.

The **makekey** command reads 10 characters from standard input and writes 13 characters to standard output. The first 8 of the 10 input characters can be any sequence of ASCII characters. The last two input characters (the *salt*), are best chosen from the set [a-zA-Z0-9.,/]. The salt characters are repeated as the first two characters of the output. The remaining 11 output characters are chosen from the same set as the salt and constitute the output key that you use as the *key* parameter to programs that perform encryption.

## Example

To generate an encryption key:

```
/usr/lib/makekey  
1234567890
```

This generates an encryption key based on the string 1234567890. The key 90y74T/NXw1U is displayed at the work station. Do *not* press **Ctrl-D** after typing the input key 1234567890 because this would end your shell session. Also, the shell prompt appears immediately after the generated key, instead of appearing on a separate line as it usually does. This is normal.

# mdrc

---

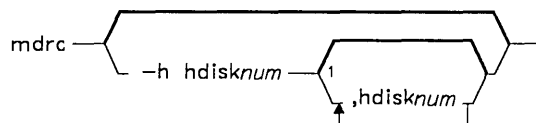
## mdrc

---

### Purpose

Allows you to reinstall a user-created minidisk after you have reinstalled AIX.

### Syntax



OL805440

### Description

The **mdrc** command provides access to user-created minidisks. You should run this command if you have reinstalled the AIX Operating System or if you have had to replace the **/etc/system**, **/etc/filesystems**, or **/etc/ddi/cpmgr** files with copies that do not contain stanzas describing any user-installed minidisks. The system uses the information in these stanzas to configure the minidisks at system startup, and **mdrc** recreates the necessary stanzas. Normally, **mdrc** uses the backup copy of **/etc/filesystems** produced by the **minidisk** command when you use it to create a new minidisk. This backup copy is named **/u/filesystems**.

If **mdrc** cannot recreate the original **/etc/filesystems** stanza for AIX Operating System minidisks, it assigns attributes of **Auto Mount = no**, **Read/Write Status = R/W**, and **Mount Directory = /tmp/directory/hdn** to the minidisk. In this case, you should then run the **minidisk** command to change the attributes to the values you want. You might also need to run the **mkdir** command to create the mount directory, if you reinstalled the entire AIX Operating System.

If minidisk has been created for use by the Personal Computer AT Coprocessor, **mdrc** will update the **/etc/ddi/cpmgr** file. If you have not installed Personal Computer AT Coprocessor Services before running **mdrc**, it creates an entry in **/etc/system**, but displays a warning message because the **/etc/ddi/cpmgr** does not exist. You must run **mdrc** again after you install the Coprocessor to be able to use the Coprocessor minidisks.

The **mdrc** command does not recognize external disks and any minidisks on them if the disks are not configured. To configure an external disk and its minidisks, see “**varyon**” on page 823.

You must have superuser authority or be a member of the system group to run the **mdrc** command.

## Flag

**-h** *hdisknum*[, *hdisknum*] . . .

Specifies any disks that have been removed or damaged and tells **mdrc** to remove the minidisk configuration entries for these disks. If you do not specify this flag and an external disk is not configured, **mdrc** ignores entries in the configuration files for the external disk's minidisks.

**Note:** If you do not have any external disks, you do not need to specify this flag.

## Files

*/etc/filesystems*  
*/etc/system*  
*/etc/ddi/cpmgr*

## Related Information

The following commands: “**minidisks**” on page 485 “**mkdir**” on page 486 and “**varyon**” on page 823.

The **filesystems** and **system** files in *AIX Operating System Technical Reference*.

## mesg

---

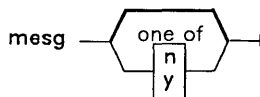
## mesg

---

### Purpose

Permits or refuses **write** messages.

### Syntax



OL805036

### Description

The **mesg** command controls whether other users on the system can send messages to you with the **write** command. Called without arguments, **mesg** displays the current work station message-permission setting. The shell start-up process permits messages by default. You can override this default action by including the line: **mesg n** in your **\$HOME/.profile** file. A user with superuser authority can send **write** messages to any work station, regardless of its message permission setting. Message permission has no effect on **mail** messages.

### Flags

- n** Disables incoming **write** messages. Use this form of the command to avoid having others clutter your display with incoming messages.
- y** Permits **write** messages.

### Files

/dev/tty\*

### Related Information

The following commands: “**write**” on page 853 and “**mail**” on page 470.

# minidisks

---

## Purpose

Adds, deletes, changes, and displays minidisks.

## Syntax

minidisks —|

OL805307

## Description

The **minidisks** command lets you add, delete, show or change characteristics of a minidisk. To use the **minidisks** command, you must be a member of the system group or have superuser authority.

The **minidisks** command is menu-driven. For information on how to use it, see *Installing and Customizing the AIX Operating System*.

## Files

/dev	Directory.
/tmp	Directory.
/etc/ddi	Directory.
/etc/master	
/etc/system	
/etc/mdkaf	
/etc/filesystems	
/tmp/CONFIGREPORT	

## Related Information

The following command: “**mkdir**” on page 486.

The discussion of **minidisks** in *Installing and Customizing the AIX Operating System*.

## **mkdir**

---

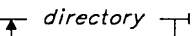
## **mkdir**

---

### **Purpose**

Makes a directory.

### **Syntax**

mkdir 

OL805037

### **Description**

The **mkdir** command makes a new *directory* in either the local or a remote node. **mkdir** creates the new directories with read, write, and execute permissions enabled for all users. You can change the permissions it sets by default with the **umask** command (see page 784.1). **mkdir** also creates by default the standard entries `.` (dot), for the directory itself, and `..` (dot dot), for its parent.

**Note:** To make a new *directory* you must have write permission in the parent directory.

### **Related Information**

The following commands: “**sh**” on page 637, “**rm**” on page 601, and “**umask**” on page 784.1.

The **mkdir** system call in *AIX Operating System Technical Reference*.

---

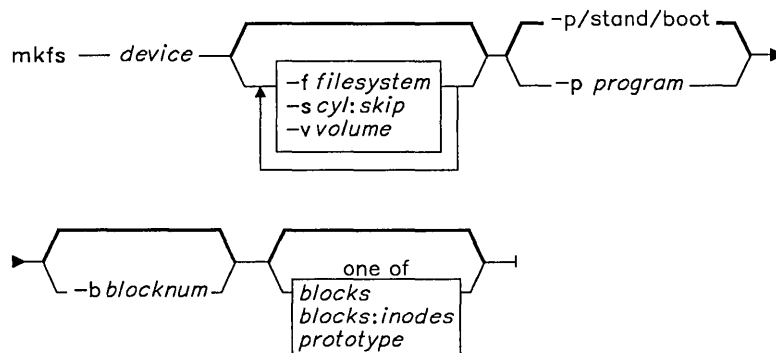
# mkfs

---

## Purpose

Makes a file system.

## Syntax



OL805364

## Description

The **mkfs** command makes new file systems. **mkfs** initializes the volume label and file system label, start-up block, bad-block list, and interleaves the free list in accordance with the flags or with defaults found in the */etc/filesystems* file.

The **mkfs** command creates the new file system on the *device* specified on the command line. *device* can be a block device name, raw device name, or file system name. If it is a file system name, **mkfs** uses this name as the *filesystem* and uses the following parameters from the applicable stanza in */etc/filesystems*:

<b>dev</b>	Device name.
<b>cyl</b>	See the following <b>-s</b> flag.
<b>skip</b>	See the following <b>-s</b> flag.
<b>vol</b>	Volume ID.
<b>bad</b>	List of bad blocks separated by commas.
<b>size</b>	File system size.
<b>boot</b>	Program to be installed in start up block.



## File System Size

You can specify the size of a new file system in the following way:

- On the command line
- In the *prototype* file
- In the `/etc/filesystems` entry for the given file system

If the size is not specified in any of these places, **mkfs** takes it from the **devinfo** structure for the block device associated with the file system being generated. (See the **ioctl** system call and the **devinfo** file in *AIX Operating System Technical Reference*.) The size provided in the **devinfo** structure is the maximum size of the file system in any case. A size specification on the command line overrides any defaults found in the **devinfo** structure or in `/etc/filesystems`.

## Prototype Files

To initialize the contents of a new file system in accordance with a prototype, specify the name of a *prototype* file on the command line. The **proto** command can be used to construct prototype files from existing file systems.

The *prototype* file contains tokens separated by spaces or new-line characters. The first token is the name of a file to be copied onto block 0 as the bootstrap program. The second token is a number specifying the size of the created file system. Typically it is the number of blocks on the device, perhaps diminished by space for paging. The next token is the number of i-nodes in the i-list. (**mkfs** rounds this to fill out the appropriate number of blocks.) The next set of tokens contains the specifications for the root file. File specifications consist of tokens giving the mode, the user name, the group name, and the initial contents of the file. The syntax of the contents field depends on the mode.

The mode token for a file is a six-character string. The first character specifies the type of the file. (The characters **-**, **b**, **c**, and **d** specify regular, block special, character special, and directory files, respectively.) The second character must be either **u** or **-**. If **u** is used, the set-user-ID mode is specified; if **-** is used the set-user-ID mode is not specified. The third character must be either **g** or **-** for specifying the set-group-ID mode. The rest of the mode is a three-digit octal number giving the owner, group, and other read, write, execute permissions (see “**chmod**” on page 128).

Two decimal number tokens come after the mode. They specify the user and group names of the owner of the file.

If the file is a regular file, the next token is a path name from which the contents and size are copied.

If the file is a block or character special file, two decimal number tokens follow, which give the major and minor device numbers.

---

If the file is a directory, **mkfs** makes the entries **.** (dot) and **..** (dot dot) and then recursively reads a list of names and file specifications for the entries in the directory. The scan is ended with the token **\$** (dollar sign).

## Flags

- bblocknum** When present, specifies the number of blocks allocated to file **i-node1** which is automatically created.
- ffilesystem** Specifies the file system label for the new file system. This can be up to six characters.
- pprogram** Specifies the name of a program to be installed in block 0 of the new file system. The default bootstrap program is **/stand/boot**.
- scyl:skip** Specifies an interleaving of the free list. (Interleaving the free list can improve the speed of disk I/O.) *cyl* is the number of blocks per cylinder, and *skip* is the number of blocks to skip.
- vvolume** Specifies the volume label for the new file system. This can be up to six characters.
- blocks[:inodes]** A size specification where *blocks* is the number of 512-byte blocks in the file system. When *inodes* is specified, it determines the number of i-nodes on the system. If *inodes* is not specified, a number suitable for the size of the file system is used. The number of i-nodes is rounded up so that the i-node area occupies an integral number of blocks.

## Examples

1. To create an empty file system on a diskette:

```
mkfs /dev/fd0
```

2. To specify volume and file system names for a new file system:

```
mkfs /dev/fd0 -fWORKFS -vVOL001
```

This creates an empty file system on the diskette, giving it the volume serial VOL001 and file system name WORKFS.

## Related Information

The following command: “**fsck**, **dfsc**” on page 333.

The **ioctl** system call and the **devinfo**, **dir**, **filesystems**, and **fs** files in *AIX Operating System Technical Reference*.

# mknod

---

## mknod

---

### Purpose

Creates a special file.

### Syntax

mknod - *device* - { *c* / *b* } - *major-minor* - *p*

OL805146

### Description

The **mknod** command makes a directory entry and corresponding i-node for a special file. The first parameter is the name of the entry *device*. Select a name that is descriptive of the device.

The **mknod** command has two forms. In the first case, the second argument is **b** or **c**. **b** indicates that the special file is a block-oriented device (disk, diskette, tape). **c** indicates that it is a character-oriented device (other devices). The last two parameters are numbers specifying the *major* device, which helps the operating system find the device driver code, and the *minor* device, that is, the unit drive, or line number, which may be either decimal or octal.

The assignment of major device numbers is specific to each system. Device numbers are determined by examining the system source file **conf.c**.

**Note:** If you change the contents of **conf.c** to add a device driver, you must rebuild the operating system. See the discussion of device drivers in *AIX Operating System Programming Tools and Interfaces* and in *AIX Operating System Technical Reference*.

The second form of **mknod** is used to create FIFOs (named pipes). The **p** flag after *device* indicates that you are creating a named pipe. See the *AIX Operating System Technical Reference* for an explanation of FIFOs and named pipes.

### Example

To create the special file for a new diskette drive:

```
mknod /dev/fd2 b 1 2
```

This creates the special file `/dev/fd2`, which is a block special file with major device number 1 and minor device number 2.

## **Related Information**

The **mknod** file and device driver description in *AIX Operating System Technical Reference*.

The discussion of device drivers in *AIX Operating System Programming Tools and Interfaces*.

## mm

---

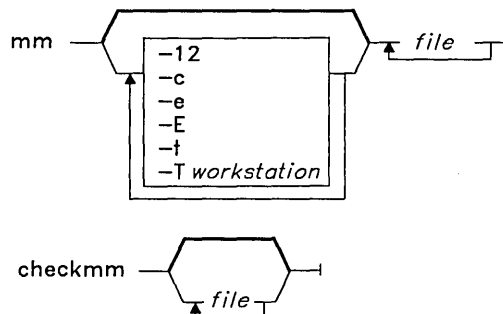
### mm, checkmm

---

#### Purpose

Displays or checks documents formatted with Memorandum Macros.

#### Syntax



OL805039

#### Description

Using the **nroff** command and the Memorandum Macro text-formatting package (MM), the **mm** command writes *files* to standard output. If you specify a - (minus) instead of any *files*, **mm** reads standard input. Do not specify both file names and standard input on the command line.

The **mm** command has flags to specify preprocessing by the **tbl** and/or **eqn** commands and postprocessing by various work station oriented output filter. It generates the proper pipelines and the required arguments for **nroff** and MM, depending on the flags selected, creates the required pipelines.

The **checkmm** command is a program for checking the contents of the named *files* for errors in the use of MM and some **eqn** and **neqn** constructions. The program skips all directories, and if you do not specify a file, **checkmm** reads standard input.

**Note:** Use the **-olist** argument of **nroff** to specify ranges of pages to be output. Note, however, that invoking **mm** with one or more of the **-e**, **-t**, and **-** minus arguments together with **nroff -olist** may cause a harmless broken pipe diagnostic if the last page of the document is not specified in *list*.

The **mm** command calls **nroff** with the **-h** flag. With this flag, **nroff** assumes that the work station has tabs set every eight character positions.

If you use the **-s** flag of **nroff** (to stop between pages of output), use a linefeed (rather than **Enter** or a new-line character) to restart the output. The **-s** flag of **nroff** does not work with the **-c** flag of **mm** or if **mm** automatically calls the **col** command.

If you provide inaccurate information to **mm** about the kind of work station its output is to be printed on, you will get unsatisfactory results; however, if you are redirecting output to a file, use the **-T37** flag and then use the appropriate work station filter when you actually print the file.

## Flags

Any flags on the command line not listed below are passed to **nroff** or to **MM**, as appropriate. The flags can occur in any order, but they must come before *file*. To obtain a list of **mm** flags, enter the command name with no arguments.

- c** Invokes the **col** command. Note that **col** is invoked automatically by **mm** unless *workstation* (the **-T** flag parameter) is one of the following:
  - 300
  - 300s
  - 450
  - 37
  - 4000a
  - 382
  - 4014
  - tek
  - 1620
  - X
- e** Invokes the **neqn** command.
- E** Invokes the **-e** flag of **nroff**.
- t** Invokes the **tbl** command.
- Tworkstation** Uses work station specification *workstation*. For a list of recognized values for *workstation*, enter:
 

```
help term1
```

By default, **mm** uses the value of the shell variable **\$TERM** from the environment as the value of *workstation*. If **\$TERM** is not set **mm** uses **lp**. If several work station types are specified, the last one listed takes effect.
- 12** Uses 12-pitch font. This may be used when **\$TERM** is set to one of **300**, **300s**, **450**, or **1620**. (The pitch switch on the DASI 300 and 300s work stations must be manually set to 12 if this flag is used.)

### Related Information

The following commands: “**col**” on page 140, “**env**” on page 298, “**eqn, neqn, checked**” on page 300, “**greek**” on page 379, “**mmt, checkmm**” on page 495, “**nroff**” on page 525, and “**tbl**” on page 739.

The **profile** file and the **eqnchar, mm, and term** miscellaneous facilities in *AIX Operating System Technical Reference*.

The discussion of **mm** in *Text Formatting Guide*.

---

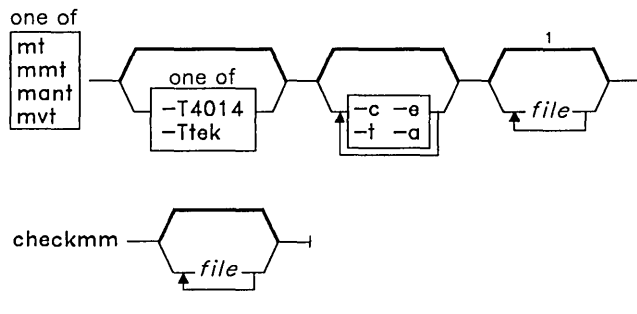
## mmt, checkmm

---

### Purpose

Typesets documents, manual pages, view graphs, and slides.

### Syntax



<sup>1</sup> If no files are given, these commands will display their flags.

OL805092

### Description

These commands are similar to the `mm` command, except they typeset their input via `troff` as opposed to formatting it via `nroff`. The `mvt`, `mt`, and `mant` commands are links to `mmt`. `mmt` uses the MM Macro Package (see `mm` in *AIX Operating System Technical Reference*), `mvt` uses the macro package for view graphs and slides (see `mv` in *AIX Operating System Technical Reference*), `mant` uses the manual page macros, and `mt` does not use a macro package.

These commands have flags to specify preprocessing by `tbl`, `cw`, or `eqn`. `mmt` generates the proper pipelines and the required arguments for `troff` and for the macro package used, depending on the flags selected. These commands read standard input if you specify a - (minus) instead of any file names.

The `checkmm` command can be used to check the input to `mmt`.

If the input contains a `troff` comment line consisting solely of the string `'\" x` (single quotation mark, backslash, double quotation mark `x`), where `x` is any combination of the three letters `c`, `e`, and `t` and where there is exactly one blank between the double quotation



mark and *x*, then the input will be processed through the appropriate combination of **cw**, **eqn**, and **tbl**, respectively, regardless of the command-line arguments.

**Note:** Use the **-olist** argument of **troff** to specify ranges of pages to be output. Note, however, that calling these commands with one or more of the **-c**, **-e**, **-t**, and **-** arguments together with **troff -olist** may cause a harmless broken pipe diagnostic if the last page of the document is not specified in *list*.

## Flags

Flags other than the ones listed below are passed to **troff** or to the macro package, as appropriate. All flags must appear before the *file* names. If you do not provide any arguments, these commands print a list of their flags.

- a** Invokes the **-a** flag of **troff**.
- c** Preprocesses the input files with **cw**.
- e** Preprocesses the input files with **eqn**.
- t** Preprocesses the input files with **tbl**.
- T4014**
- Ttek** Directs the output to a Tektronix 4014 work station via the **tc** command.

## Related Information

The following commands: “**env**” on page 298, “**eqn**, **neqn**, **checkeq**” on page 300, “**mm**, **checkmm**” on page 492, “**tbl**” on page 739, “**tc**” on page 742, and “**troff**” on page 526.

The **profile** file and the **environ**, **mm**, and **mv** miscellaneous facilities in *AIX Operating System Technical Reference*.

**moo**

---

**Purpose**

Plays a number-guessing game.

**Syntax**

`/usr/games/moo` —

OL805231

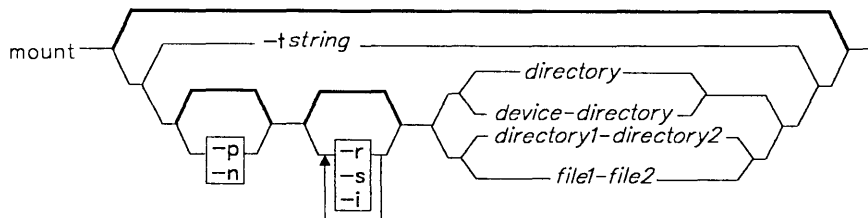
**Description**

The **moo** command picks a random four-digit decimal number with nonrepeating digits. You guess four digits and score a “cow” with a correct digit in an incorrect position and a “bull” with a correct digit in a correct position. The game continues until you guess the number.

To quit the game, press INTERRUPT (**Alt-Pause**) or END OF FILE (**Ctrl-D**).

**mount****mount****Purpose**

Makes a file system available for use.

**Syntax**

OL805467

**Description**

The **mount** command instructs the operating system to make a file system available for use. In addition, you can use **mount** to build other file trees made up directory and file mounts. In the case of file system mounts, **mount** mounts the specified *device* on the specified *directory* and records its availability in */etc/mnttab*. After **mount** has finished, *directory* becomes the root of the newly mounted file system.

Any user can issue a **mount directory directory** or **mount file file** command if :

- He has search permission to the directory or file he wants to mount (*directory1* or *file1*).
- He owns the directory or file that he wants to mount over (*directory2* or *file2*).
- He has write permission to the parent directory of *directory2* or *file2*.

Members of the system group can do any mount that a user can do plus any mount described in the */etc/filesystems* file (**mount directory**). Users operating as superuser can issue any **mount** command. Unless you specify the **-s** flag, *device* and *directory* names are restricted to strings of 100 characters or less.

The **mount** command without flags writes to standard output a list of all mounted file systems, except those mounted with the **-s** flag, along with their location and whether they are read-only. The environment variables **NLLDATE** and **NLTIME** control the appearance of the modification date and time.

If you specify only a *directory* name, **mount** takes it to be the name of the directory or file on which a file system, directory, or file is usually mounted (as defined in the */etc/filesystems* file). **mount** looks up the associated device, directory, or file and mounts

it. This is the most convenient way of using the **mount** command, as it does not require you to remember what is normally mounted on a directory or file.

The **/etc/filesystems** file should include a *stanza* for each mountable file system, directory, or file. This stanza should specify at least the name of the file system and either the device on which it resides or the directory name. If the stanza includes a *mount attribute*, the **mount** command uses the associated values. It recognizes five values in the mount attribute: **true**, **false**, **removable**, **inherit** and **read-only** (see the **filesystems** file in *AIX Operating System Technical Reference* for a description of these mount attributes.) The command **mount all** causes all file systems with the attribute **mount = true** to be mounted in their normal places. This command is typically used during system initialization.

If you are operating with superuser authority, you can mount a file system arbitrarily by naming both a *device* and a *directory* on the command line. **mount** takes *device* to be the name of the block device special file and *directory* to be the directory on which it should mount the file system.

The **mount** and **umount** programs maintain the mount table in **/etc/mnttab** as accurately as possible. However, some events can invalidate this mount table. (The system itself has an internal mount table that it maintains independently.) Several programs and library routines use **/etc/mnttab** to determine the fully qualified name of the current directory. If **/etc/mnttab** does not properly reflect the state of all mounted file systems, these programs may stop working.

Using the **-s** flag, deleting **/etc/mnttab**, or truncating it after file systems have been mounted will almost surely invalidate its contents. When the system is restarted after a crash, an old (possibly invalid) **/etc/mnttab** may still be present. If you suspect that your **/etc/mnttab** is invalid, you can use **mount** to check it and clean it up as follows:

1. Run a simple **mount** without any arguments to weed out any entries that describe file systems that are not actually mounted.
2. Enter **mount all** to remount all of the usual file systems. While **mount** cannot remount file systems that are already mounted, it notices that file systems are not listed as mounted and adds the appropriate entries to **/etc/mnttab**.

## Flags

- |                |   |
|----------------|---|
| <b>-i</b>      | Requests an inherited mount. (For information on inherited mounts, see <i>Managing the AIX Operating System</i> .)  |
| <b>-n node</b> | Specifies the node that holds the <i>directory</i> to be mounted. If you use <b>-n node</b> without a <i>directory</i> , <b>mount</b> displays a list of all mounts issued at <i>node</i> .                     |
| <b>-p</b>      | Mounts a file system as a removable file system. While there are open files, a removably mounted file system behaves the same as a normally mounted file system. However, when there are not open files (and no |

## mount

---

process has a current directory on the file system), all of the file system's disk buffers are written to the medium, and the operating system "forgets" the structure of the file system. This allows you to remove and reinsert media such as diskettes without issuing a **mount** or **umount** command each time. Use this flag only for diskette mounts.

- r** Mounts a file system as a read-only file system, regardless of the specification in **/etc/filesystems**.
- s** Does not record the availability of the new file system in **/etc/mnttab**. This allows a file systems to be mounted on a read-only root where **/etc/mnttab** would not be writable. As several programs and library routines depend on **/etc/mnttab**, use this flag with caution.
- t string** Mounts all stanzas in **/etc/filesystems** that have a **type** value equal to *string* and are not mounted.

## Examples

1. To list the file systems that are mounted:

```
mount
```

For each file system, this lists the device name, the name under which it is mounted, the access permitted (read only or read/write), and the time it was mounted.

2. To mount a diskette:

```
mount /dev/fd0 /diskette0
```

This mounts a diskette (**/dev/fd0**) onto the directory **/diskette0**. A file system must already exist on the diskette, and the directory **/diskette0** must already exist. (See "**mkfs**" on page 487 to create a file system, or "**mkdir**" on page 486 to create a directory.)

To access a file on the diskette, use a path name that begins with **/diskette0**. For example, to access **prog.c** use **/diskette0/prog.c**.

**Warning:** Be sure that the current directory is not still on the diskette when you remove it from the drive, or you may lose some of your data.

3. To mount a write-protected diskette:

```
mount -r /dev/fd0 /diskette0
```

This mounts the diskette on **/diskette0** as a read-only, file system. This tells the operating system not to update file access times, which would cause errors with a write-protected diskette.

4. To mount a default file system:

```
mount /diskette0
```

This mounts the device that is usually mounted on **/diskette0**, which is determined by information in the file **/etc/filesystems**.

5. To mount all default file systems:

```
mount all
```

This mounts all standard file systems in **/etc/filesystems** marked `mount=true`.

6. To mount a remote directory:

```
mount -n nodeA /u/tom /u/tom
```

This mounts the remote **nodeA** directory **/u/tom** onto the local node directory **/u/tom**.

7. To mount a file or directory from the **/etc.filesystem** file:

```
mount -t remote
```

This mounts all files or directories in the **/etc/filesystems** file that have a stanza that contains the attribute `type=remote`.

## Files

<b>/etc/mnttab</b>	Record of mounted file systems.
<b>/etc/filesystems</b>	Descriptions of mountable file systems.

## Related Information

The following command: “**umount**” on page 786.

The **mount**, **mntctl**, **umount**, and **vmount** system calls and the **filesystems** and **mnttab** files in *AIX Operating System Technical Reference*.

“Overview of International Character Support” in *Managing the AIX Operating System*.

## mv

---

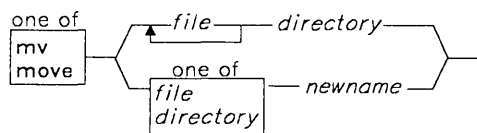
## mv

---

### Purpose

Moves files.

### Syntax



OL805010

### Description

**Warning:** The **mv** command may overwrite many files if you do not ensure that the file path names you are specifying do not already exist.

The **mv** (**move**) command moves files from one directory to another, or it renames a file or directory. If you move a *file* to a new *directory*, it retains the base file name. When you move a file, all links to other files remain intact, except when you move it to a different file system.

You can only rename a *directory* with **mv**; you cannot move it. Both *directory* and *newname* must have the same parent. All files in *directory* are moved to a newly-created directory *newname* under the same file names.

When you use **mv** to rename a *file*, then *newname* can specify either a new file name or a new directory path name. If moving the file would overwrite an existing write-protected file and if standard input is a workstation, **mv** displays the permission code of the file to be overwritten and reads one line from standard input. If the line begins with *y*, the move takes place and the file is overwritten. If not, **mv** does nothing with the *file*.

**Note:** If the *file* is on different file system than *directory*, **mv** must copy the *file* to the new file system and delete the original. In this case, the owner name becomes that of the user, and all links to other files are lost.

## Examples

1. To rename a file:

```
mv appendix apndx.a
```

This renames `appendix` to `apndx.a`. If a file named `apndx.a` already exists, its old contents are replaced with those of `appendix`.

2. To rename a directory:

```
mv book manual
```

This renames `book` to `manual`. If a directory named `manual` already exists, then an error message is displayed.

3. To move a file to another directory and give it a new name:

```
mv intro manual/chap1
```

This moves `intro` to `manual/chap1`. The name `intro` is removed from the current directory, and the same file appears as `chap1` in the directory `manual`.

4. To move a file to another directory, keeping the same name:

```
mv chap3 manual
```

This moves `chap3` to `manual/chap3`.

**Note the difference:** Examples 1 and 3 name two files, Example 2 names two existing directories, and Example 4 names a file and a directory.

5. To move several files into another directory:

```
mv chap4 jim/chap5 /u/manual
```

This moves `chap4` to `/u/manual/chap4` and `jim/chap5` to `/u/manual/chap5`.

6. To use **mv** with pattern-matching characters:

```
mv manual/* .
```

This moves all files in the directory `manual` into the current directory (`.`), giving them the same names they had in `manual`. This also empties `manual`. Note that you must type a space between the star and the period.

## Related Information

The following commands: “**chmod**” on page 128, “**ln**” on page 450, and “**rm**” on page 601.

The **rename** system call in *AIX Operating System Technical Reference*.



## **mmdir**

---

## **mmdir**

---

### **Purpose**

Moves (renames) a directory.

### **Syntax**

`mmdir — directory1 — directory2 —`

OL805137

### **Description**

The **mmdir** command renames directories within a file system. To use **mmdir**, you must have write permission to the parent directories of *directory1* and *directory2*. The *directory1* parameter must name an existing directory. If *directory2* does not exist, *directory1* is moved to *directory2*. If *directory2* exists, *directory1* becomes a subdirectory of *directory2*. Neither directory can be a subset of the other.

**Note:** *directory1* and *directory2* may be the names of files. If *directory2* is a file name, it is replaced with *directory1*.

### **Example**

To rename or move a directory to another location:

```
mmdir appendixes manual
```

If `manual` does not exist, then this renames the directory `appendixes` to `manual`. You can also rename a directory with the **mv** command.

If a directory named `manual` already exists, this moves `appendixes` and its contents to `manual/appendixes`. In other words, `appendixes` becomes a subdirectory of `manual`.

### **Related Information**

The following commands: “**mkdir**” on page 486 and “**mv**” on page 502.

The **rename** system call in *AIX Operating System Technical Reference*.

---

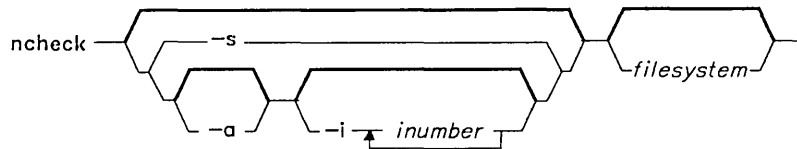
# ncheck

---

## Purpose

Generates path names from i-numbers.

## Syntax



OL805196

## Description

The **ncheck** command without any flags writes to standard output the path name and i-number list for all files in *filesystem*.

If you specify an invalid file system, the ?? in the name stands for the parent of a file system that does not have a parent. Path names beginning with ... (dot dot dot) indicate a loop.

## Flags

- a** Lists includes the file names . (dot) and .. (dot dot).
- i inumber . . .** Lists only the file specified by *inumber*.
- s** Lists only special files and files with set-user-ID mode.

## Examples

1. To list the i-number and path name of each file in the default file systems:  
ncheck

## ncheck

---

2. To list all the files in a specified file system:

```
ncheck -a /
```

This lists the i-number and path name of each file in the root file system (/), including the . (dot) and .. (dot-dot) entries in each directory (-a).

3. To list the name of a file when you know its i-number:

```
ncheck -i 690 357 280 /diskette0
```

This lists the i-number and path name for every file in the file system **/diskette0** with i-numbers of 690, 357, or 280. If a file has more than one link, all of its path names are listed.

4. To list special and set-user-ID files:

```
ncheck -s /
```

This lists the i-number and path name for every file in the root file system that is a special file (also called a *device file*) or that has set-user-ID mode enabled.

## Related Information

The following commands: “**fsck**, **dfsck**” on page 333 and “**sort**” on page 672.

## **ndtable**

---

### **Purpose**

Accesses the Distributed Services Node Table.

### **Syntax**

`ndtable`  $\rightarrow$

01.805470

### **Description**

The **ndtable** command lets you build, examine, or change the Distributed Services Network Node Table. Only members of the system group or users operating with superuser authority can use **ndtable** to change the state of the Distributed Services network node table (see “su” on page 724). Other users can use **ndtable** to browse the Network Node Table.

### **Related Information**

“Getting Started With Distributed Services Configuration Menus” in *Managing the AIX Operating System*.



---

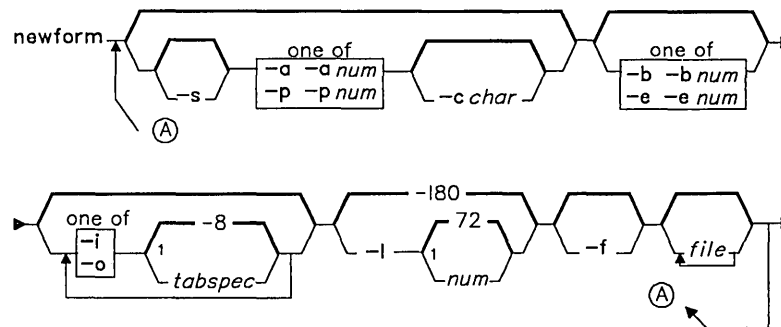
# newform

---

## Purpose

Changes the format of a text file.

## Syntax



<sup>1</sup> Do not put a blank between these items.

OL805197

## Description

The **newform** command takes lines from *file* (standard input by default), and writes the formatted lines to standard output. Lines are reformatted in accordance with command line flags in effect.

Except for **-s**, command line flags can appear in any order, can be repeated, and can be mixed with the *file* parameter. Command line flags are processed in the order specified. In other words, flag sequences like **-e15 -l60** yield results different from **-l60 -e15**. Flags are applied to all *files* on the command line.

An exit value of 0 indicates normal execution, a 1 indicates an error.

**Note:** The **newform** command normally only keeps track of physical characters; however, for the **-i** and **-o** flags, **newform** keeps track of backspaces in order to line up tabs in the appropriate logical columns.

The **newform** command does not prompt you if a *tabspec* is to be read from the standard input (by use of **-i--** or **-o --**).

If the **-f** flag is used and the last **-o** flag specified was **-o--** and was preceded by either a **-o--** or a **-i--**, the tab specification format line will be incorrect.

## newform

---

### Flags

- a**[*num*] Adds *num* characters to the end of the line when the line length is less than the effective line length (see the **-c** and **-p** flags in this section).
- b**[*num*] Truncates *num* characters from the beginning of the line when the line length is greater than the effective line length (see **-l***num*). The default action truncates the number of characters necessary to obtain the effective line length. If you specify **-b** with no *num*, the default takes effect. This flag can be used to delete the sequence numbers from a COBOL program as follows:
- ```
newform -l1-b7 file-name
```
- The **-l1** must be used to set the effective line length shorter than any existing line in the file so that the **-b** flag is activated.
- c**[*char*] Changes the prefix/add character to *char*. Default character for *char* is a space.
- e**[*num*] Same as **-b***num* except that characters are truncated from the end of the line.
- f** Writes the tab specification format line to standard output before any other lines are written. The tab specification format line displayed corresponds to the format specified in the *last* **-o** flag. If no **-o** flag is specified, the line displayed contains the default specification of **-8**.
- i**[*tabspec*] Replaces all tabs in the input with the number of spaces specified by *tabspec*. *tabspec* recognizes all tab specification forms described in “**tabs**” on page 729. If you specify a -- (minus minus) for the value of *tabspec*, **newform** assumes that the tab specification can be found in the first line read from standard input (see **fspec** in *AIX Operating System Technical Reference*). The default *tabspec* is **-8**. A *tabspec* of **-0** expects no tabs; if any are found, they are treated as **-1**.
- l**[*num*] Sets the effective line length to *num* characters. If *num* is not entered, **-l** defaults to 72. The default line length without the **-l** flag is 80 characters. Note that tabs and backspaces are considered to be one character (use **-i** to expand tabs to spaces).
- o**[*tabspec*] Replaces spaces in the input with a tab in the output, according to the tab specifications given. The default *tabspec* is **-8**. A *tabspec* of **-0** means that no spaces are converted to tabs on output.
- p**[*num*] Prefixes *num* characters (see **-c***char*) to the beginning of a line when the line length is less than the effective line length. The default action is to prefix the number of characters that are necessary to obtain the effective line length.
- s** Removes leading characters on each line up to the first tab and places up to eight of the removed characters at the end of the line. If more than eight characters (not counting the first tab) are removed, the eighth character is

replaced by an \* (asterisk) and any characters to the right of it are discarded. The first tab is always discarded.

The removed characters are saved internally until all other flags specified are applied to that line. The characters are then added at the end of the processed line.

For example, to convert a file with leading digits, one or more tabs, and text on each line, to a file beginning with the text, all tabs after the first expanded to spaces, padded with spaces out to column 72 (or truncated to column 72), and the leading digits placed starting at column 73, the command would be as follows:

```
newform -s -i -l -a -e file-name
```

The **newform** command displays an error message and stops if this flag is used on a file without a tab on each line.

## Related Information

The following commands: “**tabs**” on page 729 and “**csplit**” on page 202.

The **fspec** file in *AIX Operating System Technical Reference*.



# newgrp

---

## newgrp

---

### Purpose

Changes your primary group identification.

### Syntax

```
newgrp - [group]
```

OL805198

### Description

The **newgrp** command changes your *primary group* identification to *group*. **newgrp** recognizes only group names, not group ID numbers. Without an argument, it changes your primary group to the one specified in the */etc/passwd* file.

If the group has a password and you do not or if the group has a password and you are not listed in the */etc/group* file as a member, then **newgrp** asks you for the group password. (The use of group passwords is not encouraged because, by their very nature, they encourage poor security practices.)

**Note:** Any active user-generated shell will be terminated when **newgrp** is used.

### Flag

- Changes the environment to the login environment of the new group.

### Examples

1. To change the primary group ID of the current shell session to **admin**:  

```
newgrp admin
```
2. To change the primary group ID back to your original login group:  

```
newgrp
```

## **Files**

`/etc/group`  
`/etc/passwd`

## **Related Information**

The following commands: “**login**” on page 453 and “**users**” on page 802.

The **group** and **passwd** files in *Installing and Customizing the AIX Operating System*.

## news

---

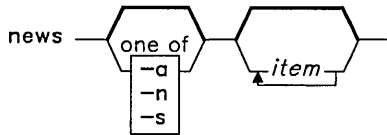
## news

---

### Purpose

Writes system news items to standard output.

### Syntax



OL805199

### Description

The **news** command keeps you informed of news concerning the system. Each news *item* is contained in a separate file in directory **/usr/news**. Anyone having read/write permission to this directory can create a news file.

If you run the **news** command without any flags, it displays every current file in **/usr/news**, showing the most recent first. Or you can specify the *items* you want displayed.

Each file is preceded by an appropriate header. To avoid reporting old news, **news** stores a currency time. **news** considers your currency time to be the modification time of the file named **\$HOME/.news\_time**. Each time you read the news, the modification time of this file changes to that of the reading. Only news item files posted after this time are considered current.

Pressing INTERRUPT (**Alt-Pause**) during the display of a news item stops the display of that item and starts the next. Pressing INTERRUPT (**Alt-Pause**) again ends **news**.

Most users run **news** each time they log in by including the line:

```
news -n
```

in their **\$HOME/.profile** file or in the system's **/etc/profile**.

### Flags

- a** Displays all news items, regardless of the currency time. The currency time does not change.

- n** Reports the names of current news items without displaying their contents. The currency time does not change.
- s** Reports the number of current news items without displaying their names or contents. The currency time does not change.

## Examples

1. To display the items that have been posted since you last read the news:

```
news
```

2. To display all the news items:

```
news -a | pg
```

This displays all the news items a page at a time (`| pg`) whether or not you have read them yet.

3. To list the names of the news items that you have not read yet:

```
news -n
```

Each name is a file in the directory `/usr/news`.

4. To display specific news items:

```
news newusers services
```

This displays news about `newusers` and `services`, which are names listed by **news -n**.

5. To display the number of news items that you have not read yet:

```
news -s
```

6. To post news for everyone to read:

```
cp schedule /usr/news
```

This copies the file `schedule` into the system news directory, `/usr/news`, to create the file `/usr/news/schedule`. To do this you must have write permission for `/usr/news`.

## Files

```
/etc/profile  
/usr/news/*  
$HOME/.news_time
```

## **Related Information**

The following command: **pg** on page 553.

The **profile** file and **environ** special facility in *AIX Operating System Technical Reference*.

---

## nice

---

### Purpose

Runs a command at a different priority.

### Syntax

```
nice number cmdstring
```

---

<sup>1</sup>Maximum increment is 19.

OL805200

### Description

The **nice** command lets you run the specified *command* at a lower priority. The value of *number* can range from 1 to 19, with 19 being the lowest priority. The default value of *number* is 10.

If you have superuser authority, you can run commands at a higher priority by specifying *number* as a negative number, such as `--10`.

### Examples

1. To run a command at low priority:

```
nice cc -c *.c
```

This runs the command `cc -c *.c` at low priority. Note that this does not run the command in the background. Your work station is not available for doing other things.

2. To run a low priority command in the background:

```
nice cc -c *.c &
```

This runs the command `cc -c *.c` at low priority in the background. Your work station is free so that you can run other commands while `cc` is running. See page 638 for details about starting background processes with `&`.

## nice

---

3. To specify a very low priority:

```
nice -15 cc -c *.c &
```

This runs **cc** in the background at a priority that is even lower than the default priority set by **nice**.

4. To specify a very high priority:

```
nice --10 wall <<end  
System shutdown in 2 minutes!  
end
```

This runs **wall** at a higher priority than all user processes. Doing this slows down everything else running on the system. If you do not have superuser authority when you run this command, then the **wall** command runs at the normal priority.

The <<end and end define a “Here Document,” which uses the text entered before the end line as standard input for the command. For more details, see “Inline Input Documents” on page 650.

## Related Information

The following command: “**nohup**” on page 523.

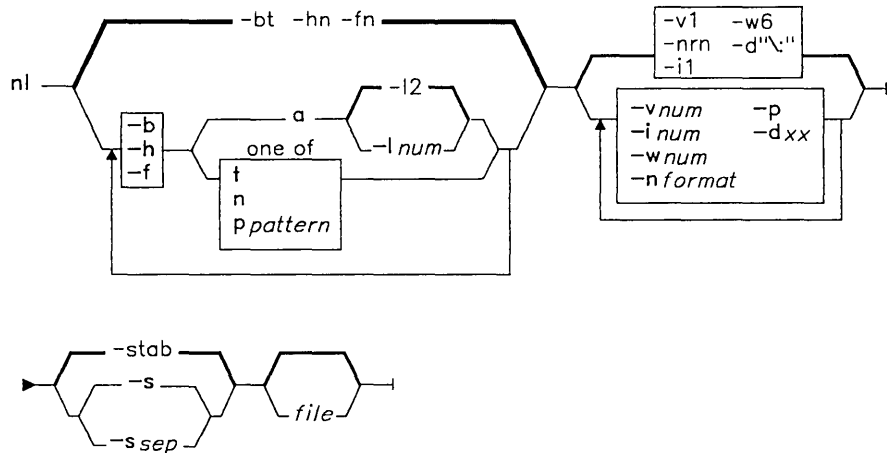
The **nice** system call in *AIX Operating System Technical Reference*.

## nl

## Purpose

Numbers lines in a file.

## Syntax



OL805386

## Description

The `nl` command reads *files* (standard input by default), numbers the lines in the input, and writes the numbered lines to standard output. In the output, `nl` numbers the lines on the left according to the flags you specify on the command line.

The input text must be written in logical pages. Each logical page has a header, a body, and a footer section (you can have empty sections). Unless you use the `-p` flag, `nl` resets the line numbers at the start of each logical page. You can set line numbering flags independently for the header, body, and footer sections (for example, no numbering of header and footer lines while numbering text lines only in the body).



Signal the start of logical page sections with lines in *file* that contain nothing but the following delimiter characters:

| <i>Line contents</i> | <i>Start of</i> |
|----------------------|-----------------|
| \: \:                | Header          |
| \: :                 | Body            |
| \:                   | Footer          |

You can name only one file on the command line. You can list the flags and the file name in any order.

## Flags

All the parameters are set by default. Use the following flags to change these default settings. Except for the **-s** flag, enter a flag without a parameter to see its default value.

- btype** Chooses which body section lines to number. The recognized *types* are:
  - a** Numbers all lines.
  - t** Does not number blank lines (default).
  - n** Does not number any lines.
  - ppattern** Numbers only those lines containing the specified *pattern*.
- dxx** Uses *xx* as the delimiters for the start of a logical page section. The default characters are \: (backslash followed by a colon). You may specify two ASCII characters, two one-byte extended characters, or one extended extended character. If you enter only one one-byte character after **-d**, the second character remains the default (colon). If you want to use a backslash as a delimiter, enter two backslashes (\\).
- ftype** Chooses which logical page footer lines to number. The *types* recognized are the same as in **-btype**. The default *type* is **n** (no lines numbered).
- htype** Chooses which logical page header lines to number. The *types* recognized are the same as in **-btype**. The default *type* is **n** (no lines numbered).
- inum** Increments logical page line numbers by *num*. The default value of *num* is 1.
- lnum** Uses *num* as the number of blank lines to count as one. For example, **-l3** will only number the third adjacent blank. The default value of *num* is 2. This flag can only be used in documents where the **-ba** flag is used.
- nformat** Uses *format* as the line numbering format. Recognized formats are:
  - ln** Left justified, leading zeroes suppressed.
  - rn** Right justified, leading zeroes suppressed (default).
  - rz** Right justified, leading zeroes kept.
- p** Does not restart numbering at logical page delimiters.

- 
- s[sep]** Separates the text from its line number by the *sep* character. The default value of *sep* is a tab character. If you enter **-s** without a parameter, there is no separation between the line number and its text.
  - vnum** Sets the initial logical page line number to *num*, (1 by default).
  - wnum** Uses *num* as the number of characters in the line number. The default value of *num* is 6.

## Examples

1. To number only the nonblank lines:

```
nl chap1
```

This displays a numbered listing of `chap1`, numbering only the nonblank lines in the body sections. If `chap1` contains no `\:\ + :`, `\:\ + :`, or `\:` delimiters, then the entire file is considered the body.

2. To number all lines:

```
nl -ba chap1
```

This numbers all the lines in the body sections, including blank lines. This form of the **nl** command is adequate for most uses.

3. To specify a different line number format:

```
nl -i10 -nrz -s:: -v10 -w4 chap1
```

This numbers the lines of `chap1` starting with ten (`-v10`) and counting by tens (`-i10`). It displays four digits for each number (`-w4`), including leading zeroes (`-nrz`). The line numbers are separated from the text by two colons (`-s::`).

For example, if `chap1` contains the text:

```
A not-so-important
note to remember:
```

```
You can't kill time
without injuring eternity.
```

then the numbered listing is:

```
0010::A not-so-important
0020::note to remember:
```

```
0030::You can't kill time
0040::without injuring eternity.
```

Note that the blank line was not numbered. To do this, use the `-ba` flag as shown in Example 2.

## **Related Information**

The following command: “**pr**” on page 561.

The “Overview of International Character Support” in *Managing the AIX Operating System*.

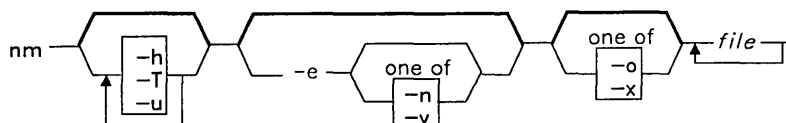
---

**nm**


---

**Purpose**

Displays the symbol table of an object file.

**Syntax**

OL805202

**Description**

The **nm** command writes the symbol table of each specified object *file* to standard output. *file* can be a single relocatable or absolute common object file or an archive library of relocatable or absolute common object files. **nm** displays the following information for each symbol:

|                |                                                                                                                                                                                                                                                                                                         |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Name</b>    | The name of the symbol.                                                                                                                                                                                                                                                                                 |
| <b>Value</b>   | Its value expressed as an offset or an address depending on its storage class.                                                                                                                                                                                                                          |
| <b>Class</b>   | Its storage class.                                                                                                                                                                                                                                                                                      |
| <b>Type</b>    | Its type and derived type. If the symbol refers to a structure or a union, the structure or union tag follows the type declaration. If the symbol is an array, the array dimensions follow the type. Note that you must have compiled the object file with <b>cc -g</b> for this information to appear. |
| <b>Size</b>    | Its size in bytes, if available. Note that you must have compiled the object file with <b>cc -g</b> for this information to appear.                                                                                                                                                                     |
| <b>Line</b>    | The source line number at which it is defined, if available. Note that you must have compiled the object file with <b>cc -g</b> for this information to appear.                                                                                                                                         |
| <b>Section</b> | For static and external storage classes, the object file section containing the symbol.                                                                                                                                                                                                                 |

## **nm**

---

### **Flag**

- e Displays only static and external symbols.
- h Does not display output header data.
- n Sorts external symbols by name before displaying them. Use this flag only in conjunction with the -e flag.
- o Displays a symbol's value and size as an octal rather than a decimal number.
- T Truncates every name that would otherwise overflow its column, making the last character displayed in the name an asterisk. By default, **nm** displays the entire name of the symbols listed, and a name that is longer than the width of the column set aside for it causes every column after the name to be misaligned.
- u Displays only undefined symbols.
- v Sorts external symbols by value before displaying them. Use this flag only in conjunction with the -e flag.
- x Displays a symbol's value and size as a hexadecimal rather than a decimal number.

### **Files**

a.out Default input file

### **Related Information**

The following commands: “**ar**” on page 58, “**as**” on page 64, “**backup**” on page 76, “**cc**” on page 112, “**ld**” on page 427, “**size**” on page 665, and “**strip**” on page 716.

The **a.out** and **ar** files in *AIX Operating System Technical Reference*.

---

# nohup

---

## Purpose

Runs a command without hangups and quits.

## Syntax

```
nohup — command —
```

OL805203

## Description

The **nohup** command runs *command*, ignoring all hangups and **QUIT** signals. Use this command to run programs in the background after you log out of the system.

Unless redirected, the output goes to the file **nohup.out**. If **nohup.out** is not writable in the current directory, the output is redirected to **\$HOME/nohup.out**

## Examples

1. To leave a command running after you log out:

```
nohup find / -print &
```

Shortly after you enter this, the following is displayed:

```
670
$ Sending output to nohup.out
```

The number will probably be different when you use this command. It is the ID of the background process started by & (ampersand). (See page 638 about starting background processes with &.) The \$ (dollar sign) is your shell prompt. Sending output . . . is a message from **nohup** telling you that it is storing the output from the `find` command in the file `nohup.out`.

You can log out after you see these messages, even if the `find` command has not finished yet.

2. To do the same, but redirecting the standard output to a different file:

```
nohup find / -print >filenames &
```

## nohup

---

This runs the `find` command and stores its output in a file named `filenames`. Now only the process ID and your prompt are displayed:

```
677  
$
```

Wait for a second or two before logging out. The **nohup** command takes a moment to start the *command* you specified. If you log out too quickly, your *command* may not run at all. Once your *command* has started, logging out will not affect it.

### Related Information

The following command: “**nice**” on page 515.

The **signal** system call in *AIX Operating System Technical Reference*.

---

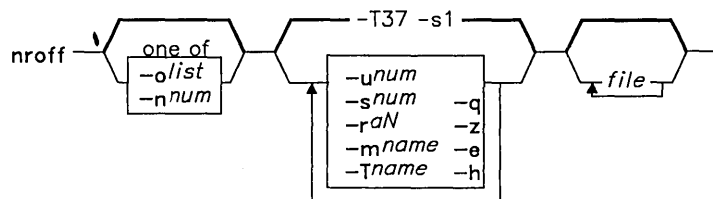
# nroff

---

## Purpose

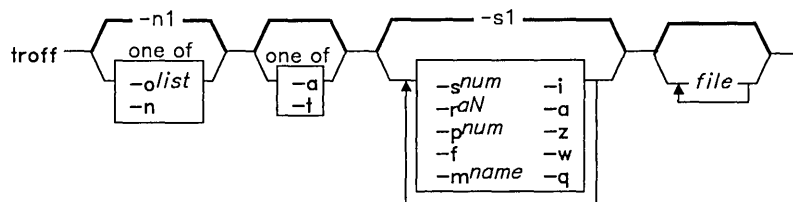
Formats text for printing devices.

## Syntax



OL805204

`troff -b`



OL805368

## Description

A complete list of **nroff** and **troff** requests, escape sequences and number registers begins on page 528. See *Text Formatting Guide* for a complete list of the naming conventions for the non-ASCII special characters and for information on writing text suitable for processing by **troff** or **nroff**.



## nroff

---

### nroff

The **nroff** command reads *files* (standard input by default), formats the text in its input for printing, and writes to standard output. **nroff** formats text for line printers and other *printing devices*, excluding phototypesetters. An input file name of - (minus) indicates standard input.

### troff

The **troff** command formats text in the input *files* (or standard input by default) for a phototypesetter, and writes its output to standard output. It is similar to the **nroff** command. An input file name of - (minus) indicates standard input.

### *nroff and troff Flags:*

- i Reads standard input after the input files.
- mname Adds `/usr/lib/tmac/tmac.name` to the beginning of the list of input file names.
- num Numbers the first printed page *num*. Do not use this flag with **-olist**.
- olist Prints only pages with page numbers appearing in *list* which consist of a comma-separated list of page numbers and ranges. A range of *A-B* means print pages *A* through *B*; an initial *-A* means print from the beginning to page *A*; and a final *A-* means print from page *A* to the end.  
  
**Note:** When this flag is used in a pipeline (for example, with **cw**, **eqn**, or **tbl**) it may cause a broken pipe diagnostic if the last page in the document is not specified in *list*.
- q Invokes the simultaneous input/output mode of the **.rd** request. **nroff** echoes the **.rd** prompt, but does not echo your input. When you enter two consecutive new-line characters, normal output is resumed.
- ra*N* Sets register *a* to *N*. *a* must have a one-character name. This is useful for automatic numbering of sections, paragraphs, lines, and so forth.
- snum Stops every *num* pages (the default is 1). **nroff** or **troff** will halt every *num* pages to allow paper loading or changing and will resume upon receipt of a line-feed or new-line character. This flag does not work in pipelines. When **nroff** halts between pages, an ASCII BEL character is sent to the printing device.
- z Suppresses the formatted output. Prints only messages generated by **.tm** (work station message) requests.

**nroff Flags:**

- e Produces equally spaced words in adjusted lines, using the full resolution of the printing device.
- h Uses tab characters during horizontal spacing. Tab settings are assumed to be every eight spaces.
- T*name* Prepares the output for the specified printing device. Known *names* are:
  - 37 TELETYPE Model 37 work station (default)
  - tn300 GE TermiNet 300 or any work station without half-line capability
  - 300s DASI 300s
  - 300 DASI 300
  - 450 DASI 450
  - lp Any ASCII line printer
  - 382 DCT-382
  - 4000A Trendata 4000A
  - 832 Anderson Jacobson 832
  - X any EBCDIC printer
  - 2631 Hewlett Packard 2631 line printer. Use a *name* of 2631-c to get compressed print. Use 2631-e to get expanded print.
- u[*num*] Sets the number of character overstrikes for boldface to *num* or to zero if *num* is not specified.

**troff Flags:**

- a Sends a printable ASCII approximation of the output to standard output.
- b Reports whether the phototypesetter is busy or available. No text processing is done.
- f Does not feed out paper and stop the phototypesetter at the end of the run.
- p*num* Prints all characters in the point size specified by *num*. Smaller point sizes may reduce the printing time.
- t Directs output without modification to standard output instead of the phototypesetter.
- w Waits until phototypesetter is available if it is currently busy.

**nroff and troff Requests**

| Request Form       | Function -- Font and Character Size Control                                          |
|--------------------|--------------------------------------------------------------------------------------|
| <b>.ps</b> $\pm N$ | Change point size by $N$ points. Also, for <b>troff</b> only, $\backslash s \pm N$ . |
| <b>.ss</b> $N$     | Space-character size set to $N/36$ em ( <b>troff</b> only).                          |
| <b>.cs</b> $F N M$ | Constant character space (width) mode (font $F$ ) ( <b>troff</b> only).              |
| <b>.bd</b> $F N$   | Embolden font $F$ by $N$ units ( <b>troff</b> only).                                 |
| <b>.bd</b> $S F N$ | Embolden Special Font when current font is $F$ ( <b>troff</b> only).                 |
| <b>.ft</b> $F$     | Change to font $F$ .                                                                 |
| <b>.fp</b> $N F$   | Mount font $F$ on position $N$ (1-4).                                                |

| Request Form       | Function -- Page Control                       |
|--------------------|------------------------------------------------|
| <b>.pl</b> $\pm N$ | Change page length by $N$ .                    |
| <b>.bp</b> $\pm N$ | Eject current page, next page number is $N$ .  |
| <b>.pn</b> $N$     | Next page number is $N$ .                      |
| <b>.po</b> $\pm N$ | Page offset = $N$ .                            |
| <b>.ne</b> $N$     | Need $N$ vertical space.                       |
| <b>.mk</b> $R$     | Mark current vertical place in register $R$ .  |
| <b>.rt</b> $\pm N$ | Return (upward only) to marked vertical place. |

| Request Form       | Function -- Text Filling, Adjusting, and Centering |
|--------------------|----------------------------------------------------|
| <b>.br</b>         | Break.                                             |
| <b>.fi</b>         | Fill subsequent output lines.                      |
| <b>.nf</b>         | No filling or adjusting of output lines.           |
| <b>.ad</b> [ $c$ ] | Adjust output lines with mode $c$ .                |
| <b>.na</b>         | Do not adjust output lines.                        |
| <b>.ce</b> $N$     | Center the following $N$ lines.                    |

| Request Form              | Function -- Vertical Spacing                                     |
|---------------------------|------------------------------------------------------------------|
| <code>.vs <i>N</i></code> | Set vertical base-line spacing to <i>N</i> .                     |
| <code>.ls <i>N</i></code> | Output <i>N</i> -1 base-line spaces after each text output line. |
| <code>.sp <i>N</i></code> | Space vertical distance <i>N</i> in either direction.            |
| <code>.sv <i>N</i></code> | Save vertical distance <i>N</i> .                                |
| <code>.os</code>          | Output saved vertical space.                                     |
| <code>.ns</code>          | Turn no-space mode on.                                           |
| <code>.rs</code>          | Restore spacing, turn no-space mode off.                         |

| Request Form                        | Function -- Line Length and Indenting            |
|-------------------------------------|--------------------------------------------------|
| <code>.li <math>\pm N</math></code> | Change line length by <i>N</i> .                 |
| <code>.in <math>\pm N</math></code> | Change indenting by <i>N</i> .                   |
| <code>.ti <math>\pm N</math></code> | Change the indent on the next line by <i>N</i> . |

| Request Form                      | Function -- Macros, Strings, Diversion, and Position Traps          |
|-----------------------------------|---------------------------------------------------------------------|
| <code>.de <i>xx yy</i></code>     | Define or redefine macro <i>xx</i> ; end at call of <i>yy</i> .     |
| <code>.am <i>xx yy</i></code>     | Append to a macro.                                                  |
| <code>.ds <i>xx string</i></code> | Define a string <i>xx</i> containing <i>string</i> .                |
| <code>.as <i>xx string</i></code> | Append <i>string</i> to string <i>xx</i> .                          |
| <code>.rm <i>xx</i></code>        | Remove request, macro, or string named <i>xx</i> .                  |
| <code>.rn <i>xx yy</i></code>     | Rename request, macro, or string <i>xx</i> to <i>yy</i> .           |
| <code>.di <i>xx</i></code>        | Divert output to macro <i>xx</i> .                                  |
| <code>.da <i>xx</i></code>        | Divert and append to <i>xx</i> .                                    |
| <code>.wh <i>N xx</i></code>      | Set location trap; negative is with respect to the end of the page. |
| <code>.ch <i>xx N</i></code>      | Change trap location.                                               |
| <code>.dt <i>N xx</i></code>      | Set a diversion trap.                                               |
| <code>.it <i>N xx</i></code>      | Set an input line trap.                                             |
| <code>.em <i>xx</i></code>        | End macro is <i>xx</i> .                                            |

## nroff

---

| Request Form                            | Function -- Number Registers                                 |
|-----------------------------------------|--------------------------------------------------------------|
| <code>.nr <math>R \pm N M</math></code> | Define and set number register $R$ ; auto-increment by $M$ . |
| <code>.af <math>R c</math></code>       | Assign format to register $R$ ( $c = 1, i, I, a, A$ ).       |
| <code>.rr <math>R</math></code>         | Remove register $R$ .                                        |

| Request Form                           | Function -- Tabs, Leaders, and Fields                              |
|----------------------------------------|--------------------------------------------------------------------|
| <code>.ta <math>Nt \dots</math></code> | Tab settings; left type, unless $t = R$ (right) or $C$ (centered). |
| <code>.tc <math>c</math></code>        | Tab repetition character.                                          |
| <code>.lc <math>c</math></code>        | Leader repetition character.                                       |
| <code>.fc <math>a b</math></code>      | Set field delimiter $a$ and pad character $b$ .                    |

| Request Form                             | Function -- Input/Output Conventions and Character Translations                     |
|------------------------------------------|-------------------------------------------------------------------------------------|
| <code>.ec <math>c</math></code>          | Set escape character.                                                               |
| <code>.eo</code>                         | Turn off escape character mechanism.                                                |
| <code>.lg <math>N</math></code>          | Ligature on if $N > 0$ .                                                            |
| <code>.ul <math>N</math></code>          | Underline in <b>nroff</b> or italicize in <b>troff</b> the next $N$ input lines.    |
| <code>.cu <math>N</math></code>          | Continuous underline in <b>nroff</b> . Acts like <code>.ul</code> in <b>troff</b> . |
| <code>.uf <math>F</math></code>          | Underline font set to $F$ (to be switched to by <code>.ul</code> ).                 |
| <code>.cc <math>c</math></code>          | Set control character to $c$ .                                                      |
| <code>.c2 <math>c</math></code>          | Set no-break control character to $c$ .                                             |
| <code>.tr <math>abcd \dots</math></code> | Translates $a$ to $b$ , and so on, on output.                                       |

| Request Form                             | Function -- Hyphenation             |
|------------------------------------------|-------------------------------------|
| <code>.nh</code>                         | No hyphenation.                     |
| <code>.hy <math>H</math></code>          | Hyphenate; $N = \text{mode}$ .      |
| <code>.hc <math>c</math></code>          | Hyphenation indicator character $c$ |
| <code>.wc <math>word \dots</math></code> | Exception words.                    |

| Request Form                         | Function -- Three Part Titles |
|--------------------------------------|-------------------------------|
| <code>.tl 'left'center'right'</code> | Three part title.             |
| <code>.pc c</code>                   | Page number character.        |
| <code>.lt ±N</code>                  | Length of title.              |

| Request Form              | Function -- Output Line Numbering      |
|---------------------------|----------------------------------------|
| <code>.nm ±N M S I</code> | Number mode on or off, set parameters. |
| <code>.nm N</code>        | Do not number next <i>N</i> lines.     |

| Request Form                                   | Function -- Conditional Acceptance of Input                                                                         |
|------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|
| <code>.if c.anything</code>                    | If condition <i>c</i> is true, accept <i>anything</i> as input. For multiple lines, use <code>\{anything\}</code> . |
| <code>.if !c anything</code>                   | If condition <i>c</i> is false, accept <i>anything</i> as input.                                                    |
| <code>.if N anything</code>                    | If expression $N > 0$ , accept <i>anything</i> as input.                                                            |
| <code>.if ! N anything</code>                  | If expression $N \leq 0$ , accept <i>anything</i> as input.                                                         |
| <code>.if 'string1' string2' anything</code>   | If <i>string1</i> is identical to <i>string2</i> , accept <i>anything</i> as input.                                 |
| <code>.if !'string1' string2.' anything</code> | If <i>string1</i> is not identical to <i>string2</i> , accept <i>anything</i> as input.                             |
| <code>.ie c anything</code>                    | If part of if-else; can take all forms of if above.                                                                 |
| <code>.el anything</code>                      | Else part of if-else.                                                                                               |

| Request Form       | Function -- Environment Switching |
|--------------------|-----------------------------------|
| <code>.ev N</code> | Environment switched (push down). |

| Request Form            | Function -- Insertions from Standard Input |
|-------------------------|--------------------------------------------|
| <code>.rd prompt</code> | Read insertion.                            |
| <code>.ex</code>        | Exit from <b>nroff</b> or <b>troff</b> .   |

## nroff

---

| Request Form             | Function -- Input/Output File Switching     |
|--------------------------|---------------------------------------------|
| <code>.so file</code>    | Switch source file (push down).             |
| <code>.nx file</code>    | Next file.                                  |
| <code>.pi program</code> | Pipe output to <i>program</i> (nroff only). |

| Request Form             | Function -- Miscellaneous                                                                                          |
|--------------------------|--------------------------------------------------------------------------------------------------------------------|
| <code>.mc c N</code>     | Set margin character <i>c</i> and separation <i>N</i> .                                                            |
| <code>.tm string</code>  | Print <i>string</i> on standard error output.                                                                      |
| <code>.ig yy</code>      | Ignore until call of <i>yy</i> .                                                                                   |
| <code>.pm t</code>       | Print macro names and sizes; if <i>t</i> is present, print only the total of sizes.                                |
| <code>.fl</code>         | Flush output buffer.                                                                                               |
| <code>.ab [text]</code>  | Prints <i>text</i> on standard error output and stops output. User abort is printed if no <i>text</i> is included. |
| <code>! cmd parms</code> | Runs the AIX command <i>cmd</i> and interpolates at that point. The standard input for <i>cmd</i> is closed.       |

## Escape Sequences for Characters, Indicators, and Functions

| Escape Sequence       | Meaning                                               |
|-----------------------|-------------------------------------------------------|
| <code>\\</code>       | Prevents or delays interpretation of <code>\</code> . |
| <code>\e</code>       | Printable version of the current escape character.    |
| <code>\.</code>       | Acute accent; equivalent to <code>\(aa</code> .       |
| <code>\`</code>       | Grave accent; equivalent to <code>\(ga</code> .       |
| <code>\-</code>       | Minus sign in the current font.                       |
| <code>\.</code>       | Dot.                                                  |
| <code>\(space)</code> | Unpaddable space-size character.                      |
| <code>\0</code>       | Digit width space.                                    |
| <code>\\</code>       | 1/6 em narrow space character (zero width in nroff).  |

| Escape Sequence              | Meaning                                                                                                           |
|------------------------------|-------------------------------------------------------------------------------------------------------------------|
| <code>\^</code>              | 1/12 em half-narrow space character (zero width in <b>nroff</b> ).                                                |
| <code>\&amp;</code>          | Non-printing, zero-width character.                                                                               |
| <code>\!</code>              | Transparent line indicator.                                                                                       |
| <code>\\$N</code>            | Interpolate argument $1 \leq N \leq 9$ .                                                                          |
| <code>\%</code>              | Default optional hyphenation character.                                                                           |
| <code>\(xx</code>            | Character named <i>xx</i> .                                                                                       |
| <code>\ *x, \ *(xx</code>    | Interpolate string <i>x</i> or <i>xx</i> .                                                                        |
| <code>\a</code>              | Non-interpreted leader character.                                                                                 |
| <code>\b'abc . . .</code>    | Bracket building function.                                                                                        |
| <code>\c</code>              | Interrupt text processing (continue word across input line break).                                                |
| <code>\d</code>              | Forward (down) 1/2 em vertical motion (1/2 line in <b>nroff</b> ).                                                |
| <code>\fx, \f(xx, \fN</code> | Change to font <i>N</i> named <i>x</i> or <i>xx</i> , or font position <i>N</i> .                                 |
| <code>\gx, \g(xx</code>      | Return the format of register <i>x</i> or <i>xx</i> . Return nothing if the register has not yet been referenced. |
| <code>\h'N'</code>           | Local horizontal motion; move right <i>N</i> (negative left).                                                     |
| <code>\jx, \jxx</code>       | Mark in register <i>x</i> or <i>xx</i> the current horizontal position on the output line.                        |
| <code>\kx</code>             | Mark horizontal input place in register <i>x</i> .                                                                |
| <code>\l'N[c]'</code>        | Horizontal line drawing function.                                                                                 |
| <code>\L'N[c]'</code>        | Vertical line drawing function.                                                                                   |
| <code>\nx, \l(xx</code>      | Interpolate number register <i>x</i> or <i>xx</i> .                                                               |
| <code>\o'abc . . . '</code>  | Overstrike characters <i>a</i> , <i>b</i> , <i>c</i> , . . . .                                                    |
| <code>\p</code>              | Break and spread output line.                                                                                     |
| <code>\r</code>              | Reverse 1 em vertical motion (reverse line in <b>nroff</b> ).                                                     |
| <code>\sN, \s±N</code>       | Point-size change function.                                                                                       |
| <code>\t</code>              | Non-interpreted horizontal tab.                                                                                   |
| <code>\u</code>              | Reverse (up) 1/2 em vertical motion (1/2 line in <b>nroff</b> ).                                                  |
| <code>\v'N'</code>           | Local vertical motion ; move down <i>N</i> (negative up).                                                         |
| <code>\w'string'</code>      | Interpolate width of <i>string</i> .                                                                              |



## nroff

---

| Escape Sequence          | Meaning                                                      |
|--------------------------|--------------------------------------------------------------|
| <code>\x'N'</code>       | Extra line-space function (negative before, positive after). |
| <code>\zc</code>         | Print <i>c</i> with zero width without spacing.              |
| <code>\{</code>          | Begin conditional input.                                     |
| <code>\}</code>          | End conditional input.                                       |
| <code>\(new line)</code> | Concealed new line.                                          |
| <code>\X</code>          | <i>X</i> , any character not listed above.                   |

## Predefined General Number Registers

| Register Name   | Description                                                     |
|-----------------|-----------------------------------------------------------------|
| <code>%</code>  | Current page number.                                            |
| <code>ct</code> | Character width type (set by width function).                   |
| <code>dl</code> | Maximum width of last completed diversion.                      |
| <code>dn</code> | Height (vertical size) of last completed diversion.             |
| <code>dw</code> | Current day of the week (1 = Sunday . . . 7 = Saturday).        |
| <code>dy</code> | Current day of the month (1-31).                                |
| <code>hp</code> | Current horizontal place on the input line.                     |
| <code>ln</code> | Output line number.                                             |
| <code>mo</code> | Current month (1-12).                                           |
| <code>nl</code> | Vertical position of last printed text base-line.               |
| <code>sb</code> | Depth of string below base line (generated by width function).  |
| <code>st</code> | Height of string above base line (generated by width function). |
| <code>yr</code> | Last two digits of current year.                                |

## Predefined Read-Only Number Registers

| Register Name | Meaning                                                                                                                               |
|---------------|---------------------------------------------------------------------------------------------------------------------------------------|
| \$            | Number of arguments available at the current macro level.                                                                             |
| .A            | Set to 1 in <b>troff</b> if the <b>-a</b> flag is used; always 1 in <b>nroff</b> .                                                    |
| .F            | The name of the current input file.                                                                                                   |
| .H            | Available horizontal resolution in basic units.                                                                                       |
| .L            | Contains the current line spacing parameter.                                                                                          |
| .P            | Contains the value 1 if the current page is being printed, 0 otherwise.                                                               |
| .R            | The number of columns available.                                                                                                      |
| .T            | Set to 1 in <b>nroff</b> , if the <b>-T</b> flag is used; always 0 in <b>troff</b> .                                                  |
| .V            | Available vertical resolution in basic units.                                                                                         |
| .a            | Post-line extra line-space most recently utilized using <code>\s'N'</code> .                                                          |
| .b            | Emboldening factor of the current font.                                                                                               |
| .c            | Number of lines read from current input file, including <b>.so</b> files.                                                             |
| .d            | Current vertical place in current diversion; equal to <b>nl</b> if no diversion.                                                      |
| .f            | Current font as physical quadrant.                                                                                                    |
| .h            | Text base-line high-water mark on current page or diversion.                                                                          |
| .i            | Current indent.                                                                                                                       |
| .j            | Current adjustment mode and type.                                                                                                     |
| .k            | Contains the horizontal size of the text portion of the current, partially-collected output line, if any, in the current environment. |
| .l            | Current line length.                                                                                                                  |
| .n            | Length of text portion on previous output line.                                                                                       |
| .o            | Current page offset.                                                                                                                  |
| .p            | Current page length.                                                                                                                  |
| .s            | Current point size.                                                                                                                   |
| .t            | Distance to the next trap.                                                                                                            |
| .u            | Equal to 1 in fill mode; equal to 0 in no-fill mode.                                                                                  |

## nroff

---

| Register Name | Meaning                              |
|---------------|--------------------------------------|
| .v            | Current vertical line spacing.       |
| .w            | Width of previous character.         |
| .x            | Reserved version-dependent register. |
| .y            | Reserved version-dependent register. |
| .z            | Name of current diversion.           |

## Files

|                      |                                                |
|----------------------|------------------------------------------------|
| /usr/lib/suftab      | Suffix hyphenation tables.                     |
| /tmp/ta\$#           | Temporary file.                                |
| /usr/lib/tmac/tmac.* | Standard macro files.                          |
| /usr/lib/macros/*    | Standard macro files.                          |
| /usr/lib/font/*      | Font width tables for <b>troff</b> .           |
| /usr/lib/term/*      | Work station driving tables for <b>nroff</b> . |

## Related Information

The following commands: “**col**” on page 140, “**cw, checkcw**” on page 213, “**eqn, neqn, checkeq**” on page 300, “**mm, checkmm**” on page 492, “**mmt, checkmmt**” on page 495, “**greek**” on page 379, “**tbl**” on page 739, and “**tc**” on page 742.

The **mm** miscellaneous facility in *AIX Operating System Technical Reference*.

The discussion of **nroff** and **troff** in *Text Formatting Guide*.

# **number**

---

## **Purpose**

Displays the written form of a number.

## **Syntax**

```
/usr/games/number ---
```

OL805229

## **Description**

The **number** game displays the written form of a number that it reads from standard input. The largest number it can translate accurately contains 66 digits.

The **number** game does not prompt you for a number. Once loaded, it simply waits for input. To exit the program, press INTERRUPT (Alt-Pause) or END OF FILE (Ctrl-D).

## **Example**

To display the written form of several numbers:

```
You: /usr/games/number
      829
System: eight hundred twenty nine.

      ...
You: 12345678
System: twelve million.
      three hundred forty five thousand.
      six hundred seventy eight.

You: Ctrl-D
```

# od

---

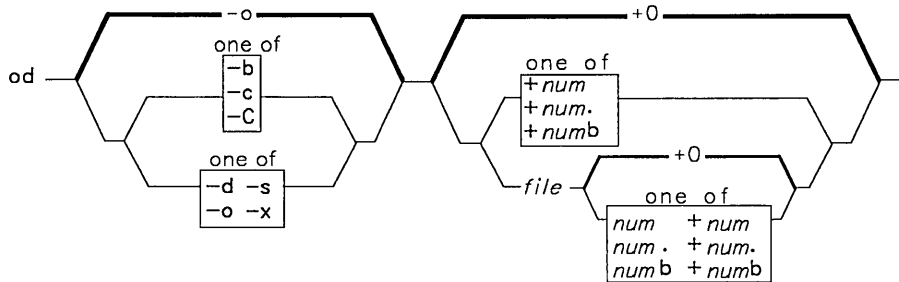
# od

---

## Purpose

Writes the contents of storage to the standard output.

## Syntax



OL805205

<sup>1</sup> Do not put a blank between these items.

OL805308

## Description

The **od** command reads *file* (standard input by default), and it writes to standard output the information stored in *file* using the format specified by the first flag. If you do not specify the first flag, **-o** is the default.

When **od** reads standard input, *num* must be preceded by a + (plus sign).

## Flags

- b** Displays bytes as octal values.
- c** Displays bytes as ASCII characters. The following nongraphic characters appear as C escapes sequences:
  - \0** Null
  - \b** Backspace
  - \f** Form feed
  - \n** New-line character

```

\r  Return
\t  Tab
\s1
\s2
\s3
\s4  Extended character shifts.

```

Others appear as 3-digit octal numbers.

- C Displays any extended characters as standard printable ASCII characters using the appropriate character escape string.
- d Displays 16-bit words as unsigned decimal values.
- o Displays 16-bit words as octal values.
- s Displays 16-bit words as signed decimal values.
- x Displays 16-bit words as hexadecimal values.

The *num* parameter specifies the point in the file where the storage output starts. The *num* parameter is interpreted as octal bytes. If a . (dot) is added to *num*, it is interpreted in decimal. If **b** is added to *num*, it is interpreted in blocks of 512 bytes.

The storage output continues until the end of the file.

## Examples

1. To display a file in octal a page at a time:

```
od a.out | pg
```

This displays **a.out** in octal (base 8) word format a page at a time.

2. To translate a file into several formats at once:

```
od -cx a.out >a.xcd
```

This writes **a.out** in hexadecimal (base 16) format (-x) into the file **a.xcd**, giving also the ASCII character equivalent, if any, of each byte (-c).

3. To start in the middle of a file:

```
od -bcx a.out +100.
```

This displays **a.out** in octal-byte, character, and hexadecimal formats, starting from the 100th byte.

**Note:** The . (dot) after the offset makes it a decimal number. Without the dot, the dump would start from the 64th (100 octal) byte.

**od**

---

## **Related Information**

The following commands: “**sdb**” on page 619 and “**pg**” on page 553.

The “Overview of International Character Support” in *Managing the AIX Operating System*.

---

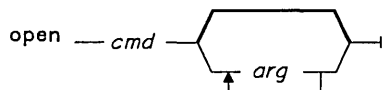
## open

---

### Purpose

Opens a virtual terminal.

### Syntax



OL805337

### Description

The **open** command opens a virtual terminal and runs the specified *file* on that terminal. If *file* does not reside in one of the directories specified in the shell **PATH** variable, you must give a full path name. Any arguments that follow *file* on the command line are passed to *file*. To move from one virtual terminal to another, press Next Window (**Alt-Action**). To close a virtual terminal, press END OF FILE (**Ctrl-D**) or end the application that is running on it.

**Note:** You should run the **actman** command before opening any virtual terminals (see “**actman**” on page 50).

### Usability Services Commands

The following additional commands are available to you from within the Usability Services Activity Manager (**/usr/bin/actmgr**):

- hide**            Removes an activity window from the window ring.
- activate**        Activates an activity window.
- cancel**           Cancels an activity window.

For details about using these commands, see *Usability Services Reference* or *Usability Services Guide*.



## open

---

### Example

To run another shell on a new virtual terminal:

```
open sh
```

To move back and forth between this new virtual terminal and any others that you have opened, press Next Window (**Alt-Action**). To close this terminal and log off the new shell, press END OF FILE (**Ctrl-D**).

### Related Information

The following command: “**actman**” on page 50.

“Using Display Station Features” in *Using the AIX Operating System*.

---

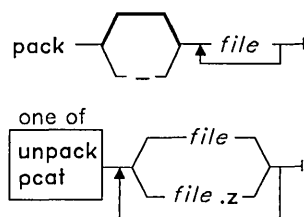
# pack

---

## Purpose

Compresses files.

## Syntax



OL805061

## Description

### pack

The **pack** command stores the specified *file* in a compressed form. The input file is replaced by a packed file with a name derived from the original file name (*file.z*), with the same access modes, access and modification dates, and owner as the original file. The input file name must contain no more than 12 characters to allow space for the added *.z* extension. Directories cannot be compressed.

If **pack** cannot create a smaller file, it stops processing and reports that it is unable to save space. (A failure to save space, generally happens with small files or files with uniform character distribution.) The amount of space saved depends on the size of the input file and the character frequency distribution. Because a decoding tree forms the first part of each *.z* file, you will generally not be able to save space with files smaller than three blocks. Typically, text files are reduced 25 to 40 percent.

The exit value of the **pack** command is the number of files that it could not pack. Packing is not done under any one of the following conditions:

- The file is already packed.
- The input file name has more than 12 characters.
- The file has links.
- The file is a directory.
- The file cannot be opened.
- No storage blocks are saved by packing.
- A file called *file.z* already exists.

## pack

---

- The **.z** file cannot be created.
- An I/O error occurs during processing.

**Note:** Both **pack** and **unpack** operate only on files ending in **.z**. As a result, when you specify a file name that does not end in **.z**, **pack** and **unpack** add the suffix and search the directory for a file name with that suffix.

### **pcat**

The **pcat** command reads the specified *files*, unpacks them, and writes them to standard output.

### **unpack**

The **unpack** is the reverse of the **pack** command. It reads the input *files*, expands them, and writes them to their original file name, the name without the **.z** suffix.

The exit value of **pcat** is the number of files it was unable to unpack. A file cannot be unpacked if any one of the following occurs:

- The file name (exclusive of the **.z**) has more than 12 characters.
- The file cannot be opened.
- The file is not a packed file.

The **unpack** command expands files created by **pack**. For each file specified, **unpack** searches for a file called *file.z*. If this file is a packed file, **unpack** replaces it by its expanded version. The **unpack** command names the new file name by removing the **.z** suffix from *file*. The new file has the same access modes, access and modification dates, and owner as the original packed file.

The exit value is the number of files the **unpack** command was unable to unpack. A file cannot be unpacked if any one of the following occurs:

- The file cannot be opened.
- The file is not a packed file.
- A file with the unpacked file name already exists.
- The unpacked file cannot be created.

**Note:** The **unpack** command writes a warning to standard output if the file it is unpacking has links. The new unpacked file has a different i-node than the packed file from which it was created. However, any other files linked to the packed file's original i-node still exist and are still packed.

## Flag

- Displays statistics about the input *files*. The statistics are calculated from a Huffman minimum redundancy code tree built on a byte-by-byte basis. Repeating - (minus) on the command line toggles this function.

## Examples

1. To compress files:

```
pack chap1 chap2
```

This compresses `chap1` and `chap2`, replacing them with files named `chap1.z` and `chap2.z`. **pack** displays the percent decrease in size for each file.

2. To display statistics about the amount of compression done:

```
pack - chap1 - chap2
```

This compresses `chap1` and `chap2` and displays statistics about `chap1`, but not about `chap2`. The first `-` (minus) turns on the statistic display, and the second turns it off.

3. To display compressed files:

```
pcat chap1.z chap2 | pg
```

This displays the compressed files `chap1.z` and `chap2.z` on the screen in expanded form, a page at a time (`| pg`). Note that **pcat** added the `.z` to the end of `chap2`, even though we did not enter it.

4. To use a compressed file without expanding the copy stored on disk:

```
pcat chap1.z | grep 'Greece'
```

This pipes the contents of `chap1.z` in its expanded form to the **grep** command. See page 638 for a discussion of piping.

5. To expand compressed files:

```
unpack chap1.z chap2
```

This expands the compressed files `chap1.z` and `chap2.z`, replacing them with files named `chap1` and `chap2`. Note that you can give **unpack** file names either with or without the `.z` suffix.

## Related Information

The following command: “**cat**” on page 109.

## **passwd**

---

## **passwd**

---

### **Purpose**

Changes your login password.

### **Syntax**

`passwd — user —`

OL805206

### **Description**

The **passwd** command establishes or changes the password associated with your login *user* name. When you enter this command, you get a prompt for the old password if one exists. Then you get two successive prompts for the new password. You must enter the same password twice for it to take effect.

Passwords can be from four to eight characters. Only the owner of the password or the superuser can change it. To change a password, the owner must know the old password. Only a user operating with superuser authority can create a null password (by removing the password entry from the */etc/passwd* file).

The system password file does not change when a new password is the same as the old one, nor does it change if the password has not aged sufficiently.

**Note:** The */etc/passwd* and */etc/opasswd* files must be on the same node.

### **Files**

*/etc/passwd*  
*/etc/opasswd*

### **Related Information**

The following commands: “**login**” on page 453 and “**users**” on page 802.

The **crypt** and **getpwent** subroutines and the **passwd** file in *AIX Operating System Technical Reference*.

---

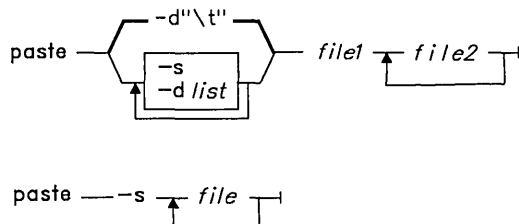
# paste

---

## Purpose

Merges the lines of several files or subsequent lines in one file.

## Syntax



OL805207

OL805366

## Description

The **paste** command reads input *files* (standard input if you specify a - as a file name), concatenates the corresponding lines of the given input files, and writes the resulting lines to standard output. Output lines are restricted to 511 characters.

Without a flag, or with the **-d** flag, **paste** treats each file as a column and joins them horizontally with a tab character by default (parallel merging). You can think of **paste** as the counterpart of the **cat** command (see page 109), which concatenates files vertically, that is, one file after another.

With the **-s** flag, **paste** combines subsequent lines of an input file (serial merging). These lines are joined with the tab character by default.

**Note:** The action of **pr -t -m** is similar to that of **paste**, but creates extra blanks, tabs and lines for a nice page layout.

## Flags

**-dlist** Changes the delimiter that separates corresponding lines in the output with one or more characters in *list* (the default is a tab). If more than one character is in *list*, then they are repeated in order until the end of the output. In parallel merging, the lines from the last file always end with a new-line character, instead of one from *list*.

## paste

---

The following special characters can also be used in *list*:

|                 |                                      |
|-----------------|--------------------------------------|
| <code>\n</code> | New-line character                   |
| <code>\t</code> | Tab                                  |
| <code>\\</code> | Backslash                            |
| <code>\0</code> | Empty string (not a null character). |
| <code>c</code>  | An extended character.               |

You must quote characters that have special meaning to the shell.

- s Merges subsequent lines from the first file horizontally. With this flag, **paste** works through one entire file before starting on the next. When it finishes merging the lines in one file, it forces a new line and then merges the lines in the next input file, continuing in the same way through the remaining input files, one at a time. A tab separates the lines unless you use the **-d** flag. Regardless of the *list*, the last character of the file is forced to be a new-line character.

## Examples

1. To paste several columns of data together:

```
paste names places dates > npd
```

This creates a file named `npd` that contains the data from `names` in one column, `places` in another, and `dates` in a third. If `names`, `places`, and `dates` look like:

| names  | places     | dates      |
|--------|------------|------------|
| rachel | New York   | February 5 |
| jerry  | Austin     | March 13   |
| mark   | Chicago    | June 21    |
| marsha | Boca Raton | July 16    |
| scott  | Seattle    | November 4 |

then `npd` contains:

```
rachel New York February 5
jerry Austin March 13
mark Chicago June 21
marsha Boca Raton July 16
scott Seattle November 4
```

A tab character separates the name, place, and date on each line. As in this example, the columns do not always line up because the tab stops are set at every eighth column.

- To separate the columns with a character other than a tab:

```
paste -d"!@" names places dates > npd
```

This alternates ! and @ as the column separators. If names, places, and dates are the same as in Example 1, then npd contains:

```
rachel!New York@February 5
jerry!Austin@March 13
mark!Chicago@June 21
marsha!Boca Raton@July 16
scott!Seattle@November 4
```

- To display the standard input in multiple columns:

```
ls | paste - - - -
```

This lists the current directory in four columns. Each - tells **paste** to create a column containing data read from the standard input. The first line is put in the first column, the second line in the second column, . . . , the fifth line in the first column, and so on.

This is equivalent to:

```
ls | paste -d"\t\t\t\n" -s -
```

which fills the columns across the page with subsequent lines from the standard input. The -d"\t\t\t\n" defines the character to insert after each column: a tab character (\t) after the first three columns, and a new-line character (\n) after the fourth. Without the -d flag, **paste -s -** would display all of the input as one line with a tab between each column.

## Related Information

The following commands: “**grep**” on page 381, “**cut**” on page 210, and “**pr**” on page 561.

The “Overview of International Character Support” in *Managing the AIX Operating System*.



# penable

---

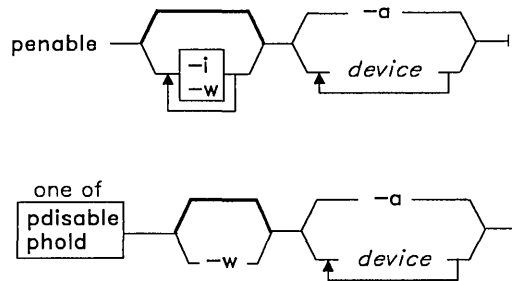
## penable

---

### Purpose

Controls or reports the availability of login ports.

### Syntax



OL805208

### Description

The **penable** command makes a port (*device*) available for logging on. If you do not specify any arguments, **penable** reports the names of all enabled ports.

Use the *device* parameter to specify the ports to be affected. Permitted values for *device* include:

- A full device name, such as `/dev/tty1`.
- A simple device name, such as `tty1`.
- A general class of devices in the form *attribute = value*, which is equivalent to naming each port with a stanza in `/etc/ports` that includes the specified attribute (see Example 4 on page 551).

These programs work by updating an entry in the `/etc/portstatus` file and then sending a signal to `init`. When `init` receives the signal and reads the updated status entry, it takes the appropriate action.

### **pdisable**

The **pdisable** command kills the logger running on the specified port, even if a user is logged on, and makes the ports unavailable for logging in. If you do not specify any arguments, **pdisable** reports the names of all disabled ports.

### **phold**

The **phold** command allows logged-on users to continue, but does not allow any more users to log on. If you do not specify any arguments, **phold** reports the names of all ports on hold.

## **Flags**

- a With **penable**, enables all ports normally enabled in the `/etc/ports` file. This is equivalent to **penable enabled = true**.  
With **pdisable** and **phold**, disables or holds all ports currently enabled in the `portstatus` file.
- i Reinitializes an existing `/etc/portstatus` file instead of updating the existing one. This flag can only be used with **penable**. You typically use this flag in the `/etc/rc` command file to re-establish default port enabling before starting up the system with multiple users.
- w Makes the applicable command return immediately rather than wait for **init** to confirm the changes in port status. You must use this flag when running **penable** either in maintenance mode or from `/etc/rc` because **init** does not initiate loggers until the system is in normal mode.

## **Examples**

1. To list the ports that are currently enabled:  
`penable`
2. To list the ports that are currently on hold:  
`phold`
3. To disable the work station attached to the `/dev/tty8` port:  
`pdisable tty8`
4. To put all 9600 baud ports on hold:  
`phold speed=9600`

## penable

---

### Files

|                              |                                          |
|------------------------------|------------------------------------------|
| <code>/etc/ports</code>      | Descriptions of all known login ports.   |
| <code>/etc/portstatus</code> | Current status of each known login port. |

### Related Information

The following command: “**init**” on page 396.

The **ports** and **portstatus** files in *AIX Operating System Technical Reference*.

---

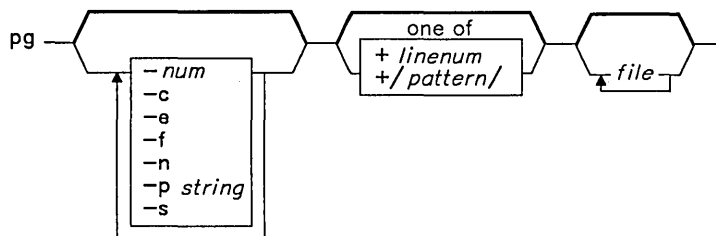
# pg

---

## Purpose

Formats files to the work station.

## Syntax



OL805245

## Description

The **pg** command reads *files* and writes them to standard output one screen at a time. If you specify *file* as - (minus) or run **pg** without arguments, **pg** reads standard input. Each screen is followed by a prompt. If you press the **Enter** key, another page is displayed. The **pg** command lets you back up to review something that has already passed.

To determine work station attributes, **pg** scans the file **terminfo** for the work station type specified by the environment variable **TERM**. The default type is **dumb**. See *AIX Operating System Technical Reference* for information on **terminfo**.

## Subcommands

When **pg** pauses and issues its prompt, you can issue a subcommand. Some of these subcommands change the display to a particular place in the file, some search for specific patterns in the text, and others change the environment in which **pg** works.

The following commands display a selected place in the file:

|              |                                                             |
|--------------|-------------------------------------------------------------|
| <i>page</i>  | Displays the specified <i>page</i> .                        |
| <i>+ num</i> | Displays the page <i>num</i> pages after the current page.  |
| <i>- num</i> | Displays the page <i>num</i> pages before the current page. |
| <b>l</b>     | Scrolls the display one line forward.                       |

|               |                                                                                                |
|---------------|------------------------------------------------------------------------------------------------|
| <i>numl</i>   | Displays a screen with the specified line <i>number</i> at the top.                            |
| + <i>numl</i> | Scrolls the display <i>num</i> lines forward.                                                  |
| - <i>numl</i> | Scrolls the display <i>num</i> lines backward.                                                 |
| <b>d</b>      | Scrolls half a screen forward. Pressing <b>Ctrl-D</b> also does this.                          |
| - <b>d</b>    | Scrolls half a screen backward. Pressing <b>-Ctrl-D</b> also does this.                        |
| <b>Ctrl-L</b> | Displays the current page again. A single period also does this.                               |
| <b>\$</b>     | Displays the last page in the <i>file</i> . Do not use this when the input is from a pipeline. |

The following commands search for text patterns in the text. You can use the patterns described in “ed” on page 280. They must always end with a new line character, even if the **-n** flag is used. In an expression such as [a-z], the minus means “through” according to the current collating sequence. A collating sequence may define *equivalence classes* for use in character ranges. See the “Overview of International Character Support” in *Managing the AIX Operating System* for more information on collating sequences and equivalence classes.

[*num*]/*pattern*/ Search for the *num*th occurrence of *pattern*. The search begins immediately after the current page and continues to the end of the current file, without wrap around. The default for *num* is 1.

*num*?*pattern*?  
*num*^*pattern*^ Search backward for the *num*th occurrence of *pattern*. The searching begins immediately before the current page and continues to the beginning of the current file, without wrap around. The ^ (circumflex) is useful for the Adds 100 work station, which cannot handle the ?. The default for *num* is 1.

After searching, **pg** normally displays the line found at the top of the screen. You can change this by adding **m** or **b** to the search command to leave the line found in the middle or at the bottom of the window with all succeeding subcommands. Use the suffix **t** to return to displaying the line with the pattern to the top of the screen.

You can change the **pg** environment with the following subcommands:

|                         |                                                                                                                                                                                 |
|-------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [ <i>num</i> ] <b>n</b> | Begins examining the <i>num</i> th next file in the command line. The default <i>num</i> is 1.                                                                                  |
| [ <i>num</i> ] <b>p</b> | Begins examining the <i>num</i> th previous file on the command line. The default <i>num</i> is 1.                                                                              |
| [ <i>num</i> ] <b>w</b> | Displays another window of text. If <i>num</i> is present, sets the window size to <i>num</i> .                                                                                 |
| <b>s</b> <i>file</i>    | Saves the input in <i>file</i> . Only the current file being examined is saved. This command must always end with a new line character, even if you specify the <b>-n</b> flag. |

- h** Displays an abbreviated summary of available subcommands.
- q or Q** Quits **pg**.
- !AIX-cmd** Sends the specified AIX command to the shell named in the **SHELL** environment variable. If this is not available, the default shell is used. This command must always end with a new line character, even if the **-n** flag is used.

At any time when output is being sent to the work station, you can press **QUIT WITH DUMP (Ctrl-V)** or **INTERRUPT (Alt-Pause)**. This causes **pg** to stop sending output and displays the prompt. Then you can enter one of the above commands in the normal manner.

**Note:** Some output is lost when when you press **QUIT WITH DUMP (Ctrl-V)** or **INTERRUPT (Alt-Pause)** because any characters waiting in the output queue are purged when the **QUIT** signal is received.

If standard output is not a work station, **pg** acts like the **cat** command, except that a header displays before each file.

While waiting for work station input, **pg** stops running when you press **INTERRUPT (Alt-Pause)**. Between prompts these signals interrupt the current task and place you in the prompt mode.

**Note:** When you use **pg** in a pipe, an Interrupt is likely to end the other commands in the pipe.

If work station tabs are not set every eight positions, unpredictable results can occur.

When using **pg** in a pipe with other commands that change work station I/O options, work station settings may not be restored correctly.

## Flags

- c** Moves the cursor to the home position and clears the screen before each page. This flag is ignored if **clear\_screen** is not defined for your work station type in the **terminfo** file.
- e** Does not pause at the end of each file.
- f** Does not split lines. Normally, **pg** splits lines longer than the screen width.
- n** Stops processing when a **pg** command letter is entered. Normally, commands must end with a new-line character.
- p string** Uses *string* as the prompt. If the *string* contains a **%d**, the **%d** is replaced by the current page number in the prompt. The default prompt is **:** (colon). If *string* contains spaces, you must quote it.
- s** Highlights all messages and prompts.

## pg

---

- + *linenum* Starts at *linenum*.
- num* Specifies the number of lines in the window. On work stations that contain 24 lines, the default is 23.
- + */pattern/* Starts at the first line that contains *pattern*.

## Files

*/usr/lib/terminfo/\**  
*/tmp/pg\**

## Related Information

The following commands: “**ed**” on page 280 and “**grep**” on page 381.

The **terminfo** file in *AIX Operating System Technical Reference*.

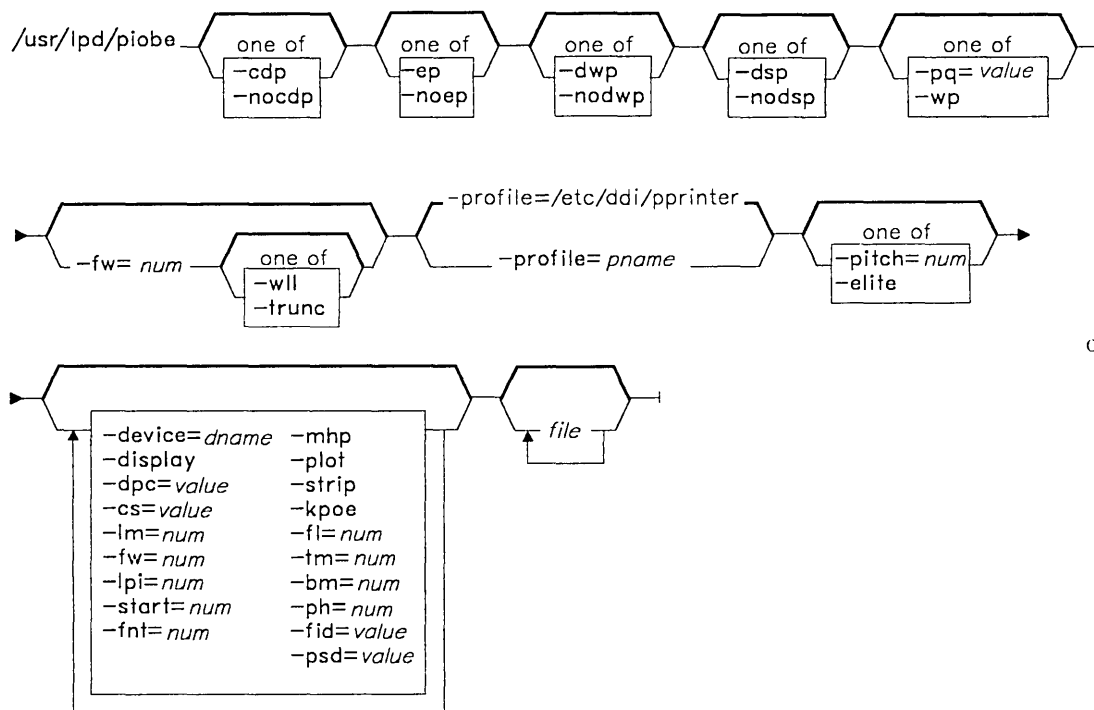
The “Overview of International Character Support” in *Managing the AIX Operating System*.

# piobe

## Purpose

Writes a file to standard output in a format suitable for sending to a line printer.

## Syntax



OL805391

OL805452

## Description

The **piobe** command writes *file* to its standard output in a form that is suitable for a line printer. If you do not specify a *file*, **piobe** reads standard input. **piobe** is normally called by the **qdaemon** command after you have enqueued a file with the **print** command (see “**print**” on page 566). The **qdaemon** directs the output from **piobe** to the appropriate device.



**piobe**

---

**Flags**

You can specify the following flags on the **print** command line or in the **/etc/qconfig** file (see *AIX Operating System Technical Reference*).

- bm = num**            Sets the bottom margin to *num* lines from the top of the page.
- cdp**
- nocdp**            Turns the condensed printing mode on (**-cdp**) or off (**-nocdp**).
- cs = value**        Uses PC code set **1** or **2**.
- device = dname**    Specifies the name of a printer stanza in the printer configuration file (see "Files" on page 559).
- display**            Specifies that the input data stream has KSR code page controls.
- dpc = value**        Prints in the specified color. Valid color *values* are **red**, **blue**, **yellow**, and **black**.
- dsp**
- nodsp**            Turns the double strike mode on (**-dsp**) or off (**-nodsp**).
- dwp**
- nodwp**            Turns the double wide printing mode on (**-dwp**) or off (**-nodwp**).
- elite**             Sets the character pitch to **12**, the same as specifying **-pitch = 12**.
- ep**
- noep**             Turns the emphasized printing mode on (**-ep**) or off (**-noep**).
- fid = value**        Specifies the font identifier for an IBM 5202 Quietwriter® III Printer font. Valid values for embedded fonts are **11** (Courier 10), **85** (Courier 12), **254** (Courier 17), and **159** (Boldface). Values for fonts in the pluggable cartridges precede the font name on the cartridge label.
- fl = num**            Sets the form length to *num*.
- fnt = num**         Allows font change. Valid values for *num* are **1** through **8**.
- fw = num**            Sets the right margin at *num* characters from the left edge of the carriage.
- kpoe**             Forgives keying mistakes and ignores invalid flags. If you specify this flag, **piobe** processes the job and sends you no message. If you do not specify this flag, **piobe** does not forgive invalid flags and does not print the job. In this case, it sends you a message detailing the error.
- lm = num**            Sets the left margin at *num* characters from the left edge of the carriage.
- lpi = num**         Sets the number of lines per inch to *num*. Valid settings are **6** and **8**.

- 
- mhp** Allows the horizontal position on the print line to be maintained for line feed and vertical tab controls, if desired.
  - ph = num** Allows you to use single-sheet paper in the Quietwriter® printer. The printer stops at the end of each page, beeps three times, and waits for you to push the start button. *num* can have the following values:
    - 0** Manual operation.
    - 1** Sheetfeed operation.
    - 2** Continuous operation.
  - pitch = num** Sets the character pitch to *num*.
  - plot** Specifies that the input data is to be passed through without modification. This allows arbitrary files to be printed on arbitrary printers.
  - pq = value** Prints in specified print quality. Valid quality *values* are **dp**, **text**, and **letter**.
  - profile = pname** Specifies the name of a printer configuration file. The default name is **/etc/ddi/pprinter**.
  - psd = value** Specifies a paper source drawer for the optional IBM 5202 Quietwriter® III Printer two-drawer sheetfeeder. Valid values are **1** (top drawer), **2** (bottom drawer), and **3** (envelopes).
  - start = num** Sets the starting page number to *num*.
  - strip** Strips all multibyte controls from the data stream. This flag is useful in filter mode in order to send data that has imbedded printer controls to a nonprinter device.
  - tm = num** Sets the top margin to *num* lines.
  - trunc** Specifies that lines exceeding the value set by **-fw** should be truncated.
  - wll** Specifies that lines exceeding the value set with the **-fw** flag should overflow to the next line. This is the reverse of the **-trunc** flag.
  - wp** Selects word processing mode, the same as specifying **-pq = letter**.

## Files

- /etc/ddi/pprinter** Parallel configuration information.
- /etc/ddi/sprinter** Serial configuration information.

### Related Information

The following commands: “**print**” on page 566 and “**qdaemon**” on page 590.

The **qconfig** file in *AIX Operating System Technical Reference*.

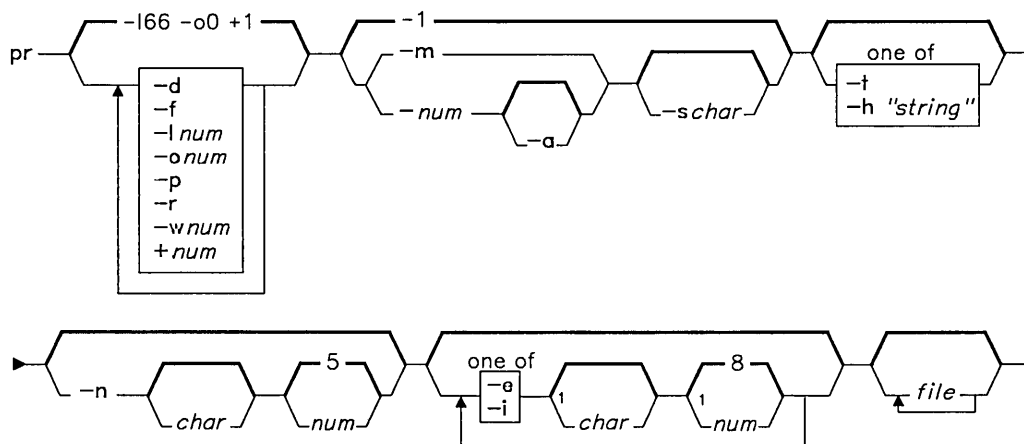
---

**pr**


---

**Purpose**

Writes a file to standard output.

**Syntax**


<sup>1</sup> Do not put a blank between these items.

OL805437

**Description**

The **pr** command writes *file* to the standard output. If you do not specify *file* or if *file* is a - (minus), **pr** reads standard input. A heading that contains the page number, date, time, and the name of the file separates the output into pages.

Unless specified, columns are of equal width and separated by at least one space. Lines that are too long for the page width are cut off. If the standard output is a work station, **pr** does not display any error messages until it has ended.

**Flags**

- a**                Displays multi-column output across the page.
- d**                Double-spaces the output.

- e[*char*][*num*]** Expands tabs to character positions  $num + 1$ ,  $2 * num + 1$ ,  $3 * num + 1$ , and so on. The default value of *num* is 8. Tab characters in the input expand to the appropriate number of spaces to line up with the next tab setting. If you specify *char* (any character other than a digit) that character becomes the input tab character. The default value of *char* is the ASCII TAB character.
- f** Uses a form-feed character to advance to a new page. (Otherwise **pr** issues a sequence of line-feed characters.) Pauses before beginning the first page if the standard output is a work station.
- h"string"** Displays *string* as the page header instead of the file name.
- i[*char*][*num*]** In the *output*, replaces white space wherever possible by inserting tabs to character positions  $num + 1$ ,  $2 * num + 1$ ,  $3 * num + 1$ , and so on. The default value of *num* is 8. If you specify *char* (any character other than a digit), that character becomes the output tab character. (The default value of *char* is the ASCII TAB character).
- l*num*** Sets the length of a page to *num* lines (the default is 66).
- m** Combines and writes all files at the same time, with each file in a separate column. (This overrides the **-num** and **-a** flags).
- n[*char*][*num*]** Provides *num*-digit line numbering (the default value of *num* is 5). The number occupies the first  $num + 1$  character positions of each column of normal output or each line of **-m** output. If you specify *char* (any character other than a digit), that character is added to the line number to separate it from whatever follows (the default value of *char* is an ASCII TAB character).
- onum** Indents each line by *num* character positions (the default is 0). The number of character positions per line is the sum of the width and offset.
- p** Pauses before beginning each page if the output is directed to a work station. (**pr** sounds the alarm at the work station and waits for you to press the **Enter** key.)
- r** Does not display diagnostic messages if the system cannot open files.
- s*char*** Separates columns by the single character *char* instead of by the appropriate number of spaces (the default for *char* is an ASCII TAB character).
- t** Does not display the five-line identifying header and the five-line footer. Stops after the last line of each file without spacing to the end of the page.
- wnum** Sets the width of a line to *num* character positions (the default value is 72 for equal-width multi-column output, no limit otherwise).
- num** Produce *num*-column output (the default is 1). The **-e** and **-i** flags are assumed for multi-column output.

---

`+ num`            Begin the display with page *num* (the default value is 1).

## Examples

1. To print a file with headings and page numbers on the printer:

```
pr prog.c | print
```

This adds page headings to `prog.c` and sends it to the **print** command. The heading consists of the date the file was last modified, the file name, and the page number.

2. To specify a title:

```
pr -h "MAIN PROGRAM" prog.c | print
```

This prints `prog.c` with the title `MAIN PROGRAM` in place of the file name. The modification date and page number are still printed.

3. To print a file in multiple columns:

```
pr -3 word.lst | print
```

This prints the file `word.lst` in three columns. Consecutive lines of `word.lst` go down the page.

4. To print several files side-by-side on the paper:

```
pr -m -h "Members and Visitors" member.lst visitor.lst | print
```

This prints `member.lst` and `visitor.lst` side by side with the title `Members and Visitors`.

5. To modify a file for later use:

```
pr -t -e prog.c > prog.notab.c
```

This replaces tab characters in `prog.c` with blanks and puts the result in `prog.notab.c`. Tab positions are at columns 9, 17, 25, 33, . . . . The `-e` tells **pr** to replace the tab characters; the `-t` suppresses the page headings.

## Files

`/dev/tty*`    To suspend messages.

## Related Information

The following command: “**cat**” on page 109.

# profiler

---

## profiler

---

### Purpose

Profiles the operating system.

### Syntax

prfld — */unix* — *kernel-image* —

prfstat — *on* — *off* —

prfdc — *file* — *minutes* — *hour* —

prfsnap — *file* —

prfpr — *file* — *cutoff* — */unix* — *kernel-image* —

OL805006

### Description

With the **prfld**, **prfstat**, **prfdc**, **prfsnap**, and **prfpr** commands, you can examine the activity of the AIX operating system.

**prfld**

Use **prfld** to initialize the recording mechanism in the system. It produces a table containing the starting address of each system subroutine as extracted from *kernel-image*.

**prfstat**

Use **prfstat** to enable or disable the sampling mechanism. **prfstat** also reveals the number of text addresses being measured.

**prfdc, prfsnap**

Use **prfdc** and **prfsnap** to collect profiler data by copying the current value of all the text address counters to a file where the data can be analyzed. **prfdc** stores the counters into *file* every specified *minutes* and turns off at *hour* (0-24). **prfsnap** collects data at the time of invocation only, adding the counter values to *file*.

**prfpr**

Use **prfpr** to format the data collected by **prfdc** or **prfsnap**. It converts each text address to the nearest text symbol (as found in *kernel-image*) and displays it if the percent activity for that range is greater than *cutoff*.

**Files**

|          |                                               |
|----------|-----------------------------------------------|
| /dev/prf | Interface to profile data and text addresses. |
| /unix    | System kernel image file.                     |

**Related Information**

The **prf** file in *AIX Operating System Technical Reference*.



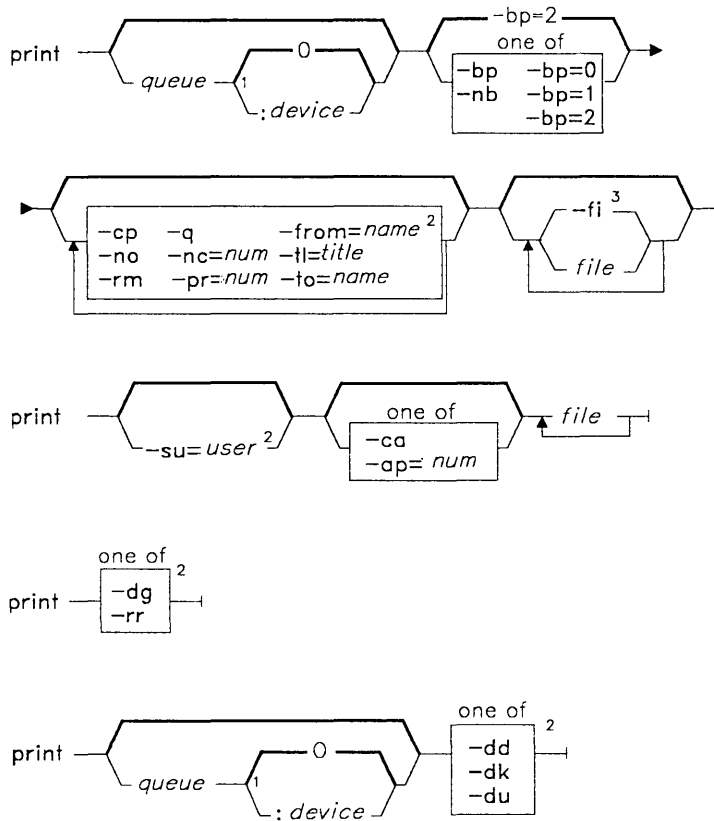
# print

# print

## Purpose

Enqueues a file.

## Syntax



OL805348

OL805147

OL805354

<sup>1</sup> Do not put a blank between these items.

<sup>2</sup> Only members of the system group can use these flags.

<sup>3</sup> Only specify `-fi` once.

## Description

With the **print** command you can enqueue files on *-queue*. (See *Managing the AIX Operating System* for information on establishing the default queue.) If more than one device services a queue, you can request a particular device by putting the device number after the queue name, separating them with a colon. Otherwise, the job is sent to the first available device.

You can also use **print** to cancel a print request (**-ca**), alter the priority of a print request (**-ap**), and display the status of print queues and devices (**-q**). Note that to print a file, you must have read access to it. To remove a file, you must have write access to the directory containing that file. If you do not specify any *file* names, **print** copies standard input into a file, which it enqueues for printing.

Flags and file names may be interspersed in any order.

**Note:** If you want to continue changing the *file* after you use **print** but before it is printed, you must use the **-cp** flag.

## Flags

If you give **print** a list of *file* names, it enqueues them all for printing on the default printer.

- ap = num** Changes to *num* the priority of the named file (which must already have been submitted for printing). See “-pr” on page 568 for a description of priorities.
- bp = num**
- bp**
- nb** Controls the printing of burst pages according to the value of *num* as follows:
  - 0** Does not print headers or trailers.
  - 1** Prints one header page before each *file*. No trailer appears.
  - 2** Prints a header page at the beginning and a trailer page at the end of each *file*.

The **header** stanza in the **qconfig** file defines the default treatment of burst pages.

Specifying only **-bp** is the same as specifying **-bp=2**. Specifying **-nb** is the same as specifying **-bp=0**.
- ca** Cancels the printing of the named *files*.
- cp** Copies the file. Ordinarily, to save disk space, **print** remembers the name of the file, but does not actually copy the file itself. Use the **-cp** flag if you want to continue changing the file while you are waiting for the current copy to be printed.

## print

---

- fi** Causes **print** to act as a filter. The **print** command automatically reads standard input if you do not specify *files* as arguments. However, if you do specify *file* arguments, you can use the **-fi** flag to force **print** to read standard input at the appropriate time.
- nc = num** Prints *num* copies of the file. Normally a file is printed only once.
- no** Notifies you when your job is finished. If the **-to** flag is also used, **print** notifies the user for whom the request is intended (see the **-to** flag on page 568).
- pr = num** Sets the priority of the named *file* to *num*. Higher numbers assign higher priority. The default priority is 15. The maximum priority is 20 for most users and 30 for the users with superuser authority and members of the system group (group 0).
- q** Displays the status of the queues and printers. The environment variable **NLTIME** controls the appearance of the **time** field.
- rm** Removes the file after it has been successfully printed.
- tl = title** Puts *title* on the header page and displays it when the **-q** flag is specified. Normally the job title is the name of the file. If **print** reads from standard input, the job title is PRIMARY.OUTPUT.
- to = name** Labels the output for delivery to *name*. Normally the output is labeled for delivery to the person issuing the print request.

In addition the flags available to all users, users with superuser authority and members of the system group (group 0), can use the following additional flags:

- dd** Turns off the device associated with *queue*. The **qdaemon** no longer sends jobs to the device, and entering **print -q** shows its status as OFF. Any job currently running on the device is allowed to finish.
- dg** Kills the **qdaemon** after all currently running jobs are finished. This is the only clean way to bring the **qdaemon** down. Using the **kill** command may cause problems, such as jobs hanging up in the queue.
- dk** Acts the same as **-dd**, except current jobs are killed. They remain in the queue, and are run again when the device is turned on.
- du** Turns on the device associated with *queue*. The **qdaemon** sends jobs to it again and entering **print -q** shows its status as READY.  
  
**Note:** If more than one device is associated with a queue, you must specify the device as well as the queue when you use the **-dd**, **-dk**, and **-du** flags. Devices are numbered, starting at zero, in the order that they appear in the **qconfig** file. For example, **-lp:0** designates the first device on the **lp** queue. **-lp** designates the same device only if there is no other device on that queue.

- from = name** Labels the output as though *name* had submitted it. You can only use this flag with superuser authority.
- rr** Forces the **qdaemon** to reread the **qconfig** file after all currently running jobs are finished. With this flag, a user with superuser authority can change the **qconfig** file without having to kill and restart the **qdaemon**.
- su = user** Cancels or changes the priority on another *user*'s job when used with the **-ca** or the **-ap** flags. For example, a job report submitted by user *ann* can be cancelled as follows:

```
print -su=ann -ca report
```

The **print** command passes flags it does not recognize to the backend that does the printing. Thus, for each queue there are flags not described above that can be included on the **print** command line. See “**piobe**” on page 557 for a list of these flags.

## Examples

1. To print a file on the default printer:

```
print memo
```

2. To print a file with page numbers:

```
pr prog.c | print
```

The **pr** command puts a heading at the top of each page that includes the date the file was last modified, the name of the file, and the page number. The **print** command then prints the file.

3. To see if a file is still waiting to be printed:

```
print -q
```

This displays the status of the queues and printers. If a file has not been printed yet, then it appears in the queue status listing. If you piped data to **print**, as in Example 2, then it is listed as PRIMARY.OUTPUT.

4. To stop printing a file:

```
print -ca chapter1
```

This cancels the request you made earlier to print the file *chapter1*. If the file is currently being printed, then the printer stops immediately. If the file has not been printed yet, then it is removed from the queue so that it will not be printed.

## print

---

If the file is not in the queue, **print** displays the message:  
no such request from you -- perhaps it's done?

5. To disconnect a printer from the queueing system:

```
print -a:2 -dd
```

This stops print requests from being sent to the third printer that serves the **-a** queue. If a file is currently being printed, it is allowed to finish. You must be a member of the **system** group (group 0) to run this command.

**Note:** The printers serving a given queue are numbered starting with zero in the order that they appear in the **/etc/qconfig** file.

## Files

|                                   |                                           |
|-----------------------------------|-------------------------------------------|
| <code>/etc/qdaemon</code>         | Queueing demon.                           |
| <code>/usr/lpd/qdir/*</code>      | Queue requests.                           |
| <code>/usr/lpd/stat/*</code>      | Information on the status of the devices. |
| <code>/usr/spool/qdaemon/*</code> | Temporary copies of enqueued files.       |
| <code>/etc/qconfig</code>         | Queue configuration file.                 |

## Related Information

The following commands: “**piobe**” on page 557, “**pr**” on page 561, and “**qdaemon**” on page 590.

The **qconfig** file in *AIX Operating System Technical Reference*.

The discussion of **print** in *Managing the AIX Operating System*.

The “Overview of International Character Support” in *Managing the AIX Operating System*.

---

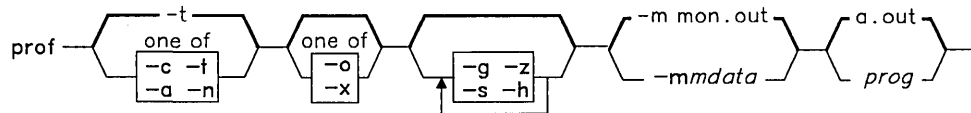
# prof

---

## Purpose

Displays program profile data.

## Syntax



OL805438

## Description

The **prof** command interprets profile data collected by the **monitor** subroutine for the object file **prog** (**a.out** by default). It reads the symbol table in the object file *prog* and correlates it with the profile file (**mon.out** by default). **prof** displays, for each external text symbol, the percentage of execution time spent between the address of that symbol and the address of the next, the number of times that function was called, and the average number of milliseconds per call.

For the number of calls to a function to be tallied, you must have compiled the file using the **-p** flag of the **cc** command. This flag also arranges for the object file to include a special profiling start-up function that calls the **monitor** subroutine at the beginning and end of execution. It is the call to **monitor** at the end of execution that writes **mon.out**. Thus, only programs that explicitly **exit** or **return** from **main** cause the **mon.out** file to be produced.

**Note:** No more than 600 functions can have call counters established during program execution. If you exceed this limit, **prof** overwrites other data and damages the **mon.out** file. **prof** automatically reports the number of call counters used whenever the number exceeds 500.

## Flags

The mutually exclusive flags **a**, **c**, **n**, and **t** determine how **prof** sorts the output lines:

- a**           Sorts by increasing symbol address.
- c**           Sorts by decreasing number of calls.
- n**           Sorts lexically by symbol name.

## prof

---

**-t** Sorts by decreasing percentage of total time (default).

The mutually exclusive flags **o** and **x** specify how to display the address of each symbol monitored.

**-o** Displays each address in octal, along with the symbol name.

**-x** Displays each address in hexadecimal, along with the symbol name.

Use the following flags in any combination:

**-g** Includes nonglobal symbols (static functions).

**-h** Suppresses the heading normally displayed on the report. (This is useful if the report is to be processed further.)

**-m** *mdata* Takes profiling data from *mdata* instead of **mon.out**.

**-s** Displays a summary of monitoring parameters and statistics on standard error.

**-z** Includes all symbols in the profile range, even if associated with zero calls and zero time.

## Files

|         |                      |
|---------|----------------------|
| mon.out | Default profile.     |
| a.out   | Default object file. |

## Related Information

The following commands: “**cc**” on page 112 and “**nm**” on page 521.

The **exit** and **profil** system calls and the **monitor** subroutine in *AIX Operating System Technical Reference*.

---

## proto

---

### Purpose

Constructs a prototype file for a file system.

### Syntax

`/etc/proto` — *directory* —  —

OL805007

### Description

The **proto** command makes a prototype file for a file system or part of a file system. Use the prototype file as input to the **mkfs** command to construct a file system according to a predefined template. The prototype file consists of a recursive directory listing of every file on the file system, with its owner, group, and protection. It also contains the file from which the prototype file is to be initialized, formatted as described in the **mkfs** command.

Specify the base directory from which the prototype file is made with *directory*. The prototype file includes the complete subtree below *directory* that is contained on the same file system as *directory*.

The *prefix* parameter is added to the names of all the initialization files, forcing the initialization files to be taken from a place other than the prototype. Before the output from **proto** can be used with **mkfs**, **mkfs** needs a start up program, a file system size, and an i-list size. Link information is not preserved with the **proto** command.

The collating sequence is determined by the **ct\_collate** array in the **NLctab** subroutine.

### Related Information

The following command: “**mkfs**” on page 487.

The “Overview of International Character Support” in *Managing the AIX Operating System*.



## prs

---

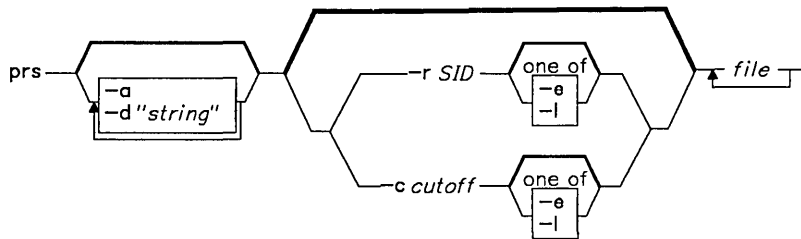
## prs

---

### Purpose

Displays a Source Code Control System (SCCS) file.

### Syntax



OL805248

### Description

The `prs` command reads *files*, and writes to standard output a part or all of a **Source Code Control System (SCCS)** file. If you specify a directory in place of *file*, `prs` performs the requested actions on all SCCS files (those with a name that has the *s.* prefix). If you specify a `-` (minus) in place of *file*, `prs` reads standard input and interprets each line as the name of an SCCS file. `prs` continues to take input until it reads an end-of-file character (Ctrl-D).

#### Data Keywords

Data keywords specify which parts of an SCCS file are to be retrieved and written to standard output. All parts of an SCCS file have an associated data keyword. There is no limit to the number of times a data keyword can appear in a *string*. The information that `prs` displays consists of user-supplied text and appropriate values (extracted from the SCCS file) substituted for the recognized data keywords in the order of appearance in *string*. The format of a data keyword value is either simple, in which the keyword substitution is direct, or multi-line, in which the substitution is followed by a carriage return. Text is any characters other than recognized data keywords. Specify a tab character with `\t` and a carriage return/new-line character with a `\n`. Remember to quote the `\t` and `\n` with an extra `\` to prevent shell from interpreting the `\` and just passing a `t` or `n` to `prs` as text.

The following table lists the keywords associated with information in the delta table in the SCCS file (see the `sccsfile` file in *AIX Operating System Technical Reference* for the structure of an SCCS file).

| Keyword | Data Represented                                           | Value                     | Format |
|---------|------------------------------------------------------------|---------------------------|--------|
| :R:     | Release number                                             | num                       | Simple |
| :L:     | Level number                                               | num                       | Simple |
| :B:     | Branch number                                              | num                       | Simple |
| :S:     | Sequence number                                            | num                       | Simple |
| :I:     | SCCS ID string (SID)                                       | :R::L::B::S:              | Simple |
| :Dy:    | Year delta created                                         | YY                        | Simple |
| :Dm:    | Month delta created                                        | MM                        | Simple |
| :Dd:    | Day delta created                                          | DD                        | Simple |
| :D:     | Date delta created                                         | YY/MM/DD                  | Simple |
| :Th:    | Hour delta created                                         | HH                        | Simple |
| :Tm:    | Minute delta created                                       | MM                        | Simple |
| :Ts:    | Second delta created                                       | SS                        | Simple |
| :T:     | Time delta created                                         | HH:MM:SS                  | Simple |
| :DT:    | Delta type                                                 | D or R                    | Simple |
| :P:     | User who created the delta                                 | login name                | Simple |
| :DS:    | Delta sequence number                                      | num                       | Simple |
| :DP:    | Previous delta sequence number                             | num                       | Simple |
| :Dt:    | Delta information                                          | :DT::I::D: :T::P::DS::DP: | Simple |
| :Dn:    | Sequence numbers of deltas included                        | :DS: . . .                | Simple |
| :Dx:    | Sequence numbers of deltas excluded                        | :DS: . . .                | Simple |
| :Dg:    | Sequence numbers of deltas ignored                         | :DS: . . .                | Simple |
| :DI:    | Sequence numbers of deltas included, excluded, and ignored | :Dn:/:Dx:/:Dg:            | Simple |
| :Li:    | Lines inserted by Delta                                    | num                       | Simple |
| :Ld:    | Lines deleted by Delta                                     | num                       | Simple |

Figure 3 (Part 1 of 2). Delta Table Keywords

| <b>Keyword</b> | <b>Data Represented</b>  | <b>Value</b>          | <b>Format</b> |
|----------------|--------------------------|-----------------------|---------------|
| <b>:Lu:</b>    | Lines unchanged by Delta | num                   | Simple        |
| <b>:DL:</b>    | Delta line statistics    | <b>:Li:/:Ld:/:Lu:</b> | Simple        |
| <b>:MR:</b>    | MR numbers for delta     | text                  | Multi-line    |
| <b>:C:</b>     | Comments for delta       | text                  | Multi-line    |

**Figure 3 (Part 2 of 2). Delta Table Keywords**

The following table lists the keywords associated with the header flags in the SCCS file. For more information of Header flags, see Figure 1 on page 54.

| <b>Keyword</b> | <b>Data Represented</b>        | <b>Value</b>     | <b>Format</b> |
|----------------|--------------------------------|------------------|---------------|
| <b>:Y:</b>     | module type                    | text             | simple        |
| <b>:MF:</b>    | MR validation flag set         | yes or no        | Simple        |
| <b>:MP:</b>    | MR validation program name     | text             | Simple        |
| <b>:KF:</b>    | Keyword/error warning flag set | yes or no        | Simple        |
| <b>:BF:</b>    | Branch flag set                | yes or no        | Simple        |
| <b>:J:</b>     | Joint edit flag set            | yes or no        | Simple        |
| <b>:LK:</b>    | Locked releases                | <b>:R: . . .</b> | Simple        |
| <b>:Q:</b>     | User defined keyword           | text             | Simple        |
| <b>:M:</b>     | Module name                    | text             | Simple        |
| <b>:FB:</b>    | Floor boundary                 | <b>:R:</b>       | Simple        |
| <b>:CB:</b>    | Ceiling boundary               | <b>:R:</b>       | Simple        |
| <b>:Ds:</b>    | Default SID                    | <b>:I:</b>       | Simple        |
| <b>:ND:</b>    | Null Delta flag set            | yes or no        | Simple        |
| <b>:FL:</b>    | Header flag list               | text             | Multi-line    |

**Figure 4. Header Flag Keywords**

The following table lists the keywords associated with other parts of the SCCS file.

| Keyword | Data Represented               | Value           | Format     |
|---------|--------------------------------|-----------------|------------|
| :UN:    | user names                     | text            | Multi-line |
| :FD:    | descriptive text               | text            | Multi-line |
| :BD:    | body of text                   | text            | Multi-line |
| :GB:    | text in a g-file               | text            | Multi-line |
| :W:     | a <b>what</b> string           | :Z::M: \tab :I: | Simple     |
| :A:     | a <b>what</b> string           | :Z::Y::M::I::Z: | Simple     |
| :Z:     | a <b>what</b> string delimiter | @(#)            | Simple     |
| :F:     | SCCS file name                 | text            | Simple     |
| :PN:    | SCCS file path name            | text            | Simple     |

Figure 5. Other Keywords

## Flags

Each flag or group of flags applies independently to each named file.

- a        Writes information for the specified deltas, whether or not they have been removed (see “**rmdel**” on page 604). If you do not specify the **-a** flag, **prs** supplies information only for the specified deltas that have not been removed.
- ccutoff    Specifies a *cutoff* date and time for the **-e** and **-l** flags. Specify *cutoff* in the following form:  
           YY[MM[DD[HH[MM[SS]]]]]]  
           All omitted items default to their maximum values, so specifying **-c8402** is the same as specifying **-c840229235959**. You can separate the fields with any non-numeric characters. For example, you can specify **-c84/2/20,9:22:25** or **-c"84/2/20 9:22:25"** or **"-c84/2/20 9:22:25"**.
- d"*string*"    Specifies the data items to be displayed. *string* is a string consisting of optional text and SCCS file data keywords. You must enclose all text and spaces in *string* in quotation marks.
- e        Requests information for all deltas created *earlier* than and including the delta specified by the **-r** flag.
- l        Requests information for all deltas created *later* than and including the delta specified by the **-r** flag.

## prs

---

**-rSID** Specifies the *SID* of a delta for which **prs** will retrieve information. If no *SID* is specified, **prs** retrieves the information for the *SID* of the highest numbered delta.

## Files

/tmp/pr?????

## Related Information

The following commands: “**admin**” on page 51, “**delta**” on page 236, “**get**” on page 359, and “**help**” on page 391.

The **sccsfile** file in *AIX Operating System Technical Reference*.

The discussion of SCCS in *AIX Operating System Programming Tools and Interfaces*.



**ps**

---

I Intermediate  
Z Canceled  
T Stopped  
K Available kernel process  
X Growing.

**UID (f,l)**

The user ID of the process owner; the login name is displayed with the **-f** flag.

**PID (all)**

The process ID of the process.

**PPID (f,l)**

The process ID of the parent process.

**C (f,l)**

Processor utilization for scheduling.

**STIME (f)**

Starting time of the process. The **NLLDATE** and **NLTIME** environment variables control the appearance of this field.

**PRI (l)**

The priority of the process; higher numbers mean lower priority.

**NI (l)**

Nice value; used in calculating priority.

**ADDR (l)**

The segment number of the process stack, if normal; if a kernel process, the address of the pre-process data area.

**SZ (l)**

The size in blocks of the core image of the process.

**WCHAN (l)**

The event for which the process is waiting or sleeping; if blank, the process is running.

**TTY (all)**

The controlling work station for the process.

**TIME (all)**

The total execution time for the process.

**CMD (all)**

The command name; the full command name and its parameters are displayed with the **-f** flag.

A process that has exited and has a parent, but has not yet been waited for by the parent, is marked **<defunct>**.

With the **-f** flag, **ps** determines what the command name and parameters were when the process was created by examining memory or the paging area. If it cannot find this information, the command name, as it would appear without the **-f** flag, displays in square brackets.

**Note:** Things can change while **ps** is running.

Some data displayed for defunct processes are irrelevant.

The current work station is defined as the one associated with standard error. Thus redirecting standard error, for example:

```
ps 2> /dev/null
```

does not work as expected.

## Flags

- a** Writes to standard output information about all processes except the process group leaders and processes not associated with a terminal.
- c** *corefile* Uses *corefile* instead of the default **/dev/mem**. *corefile* is a core image file that has been created by the **Ctrl-(left)Alt-End** key sequence.
- d** Writes information to standard output about all processes except the process group leaders.
- e** Writes information to standard output about all processes except kernel processes.
- f** Generates a full listing. The meaning of columns in a full listing is described on page 579.
- g** *glist* Writes information to standard output only about processes that are in the process groups listed in *glist*. The *glist* is either a comma-separated list of process-group identifiers or a list of process-group identifiers enclosed in double quotation marks (" ") and separated from one another by a comma and/or one or more spaces.
- k** Writes information to standard output about kernel processes. Otherwise, it does not list kernel processes.
- l** Generates a long listing. The meaning of a long listing is described on page 579.
- n** *kernel-image* Takes *kernel-image* as the name of an alternate *kernel-image* file (**/unix** is the default).
- p** *plist* Displays only information about processes with the process numbers specified in *plist*. *plist* is either a comma-separated list of process-ID numbers or a list of process-ID numbers enclosed in double quotation



- marks (" ") and separated from one another by a comma and/or one or more spaces.
- t *tlist*** Displays only information about processes associated with the work stations listed in *tlist*. *tlist* is either a list of comma-separated work-station identifiers or a list of work-station identifiers enclosed in double quotation marks (" ") and separated from one another by a comma and/or one or more spaces.
- u *ulist*** Displays only information about processes with the user ID numbers or login names specified in *ulist*. *ulist* is either a comma-separated list of user ID's or a list of user ID's enclosed in double quotation marks (" ") and separated from one another by a comma and/or one or more spaces. In the listing, **ps** displays the numerical user ID unless the **-f** flag is used; then it displays the login name.

## Examples

1. To list the processes that you have started:  

```
ps
```

This displays a short listing of information about the processes associated with your work station.
2. To display all process information available:  

```
ps -e -f -l
```

This displays all the information **ps** has to offer (**-l -f**) about all processes (**-e** for "everything").
3. To list processes owned by specific users:  

```
ps -f -l -ujim,jane,su
```

This displays all the information available (**-l -f**) about the processes being run by the users **jim**, **jane**, and **su**.
4. To list processes associated with specific work stations:  

```
ps -t-,console
```

This displays information about processes not connected to any work station (**-t-**), and processes associated with the work station **/dev/console**.

## Files

|              |                                              |
|--------------|----------------------------------------------|
| /unix        | System kernel image.                         |
| /dev/mem     | Memory.                                      |
| /etc/passwd  | Supplies UID information.                    |
| /etc/ps_data | Internal data structure.                     |
| /dev         | Searched to find work station ("TTY") names. |

## Related Information

The following commands: “**kill**” on page 422 and “**nice**” on page 515.

# ptx

---

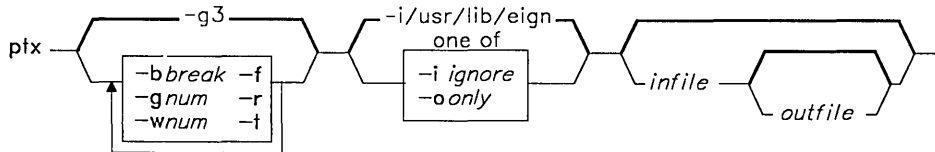
## ptx

---

### Purpose

Generates a permuted index.

### Syntax



OL805250

### Description

The **ptx** command reads *infile* (standard input by default), creates a permuted index from its input, and writes to *outfile* (standard output by default).

The **ptx** command searches *infile* for keywords, sorts the lines, and generates the file *outfile*. *outfile* can then be processed with **nroff** or **troff** to produce a permuted index from the file *infile*.

The **ptx** command follows three steps:

1. In the permutation, generates one line for each keyword in an input line, and rotates the keyword to the front.
2. Sorts the permuted file.
3. Rotates the sorted lines so that the keyword comes at the middle of each line.

The resulting lines in output are in the form:

```
.xx, "tail" "before keyword" "keyword and after" "head"
```

where **.xx** is an **nroff** or **troff** macro provided by the user, or provided by the **mptx** macro package (see the *AIX Operating System Technical Reference* for information on this macro package). The *before keyword* and *keyword* and *after* fields incorporate as much of the line as will fit around the keyword when it is printed. *tail* or *head*, at least one of which is always the empty string, are wrapped-around pieces small enough to fit in the unused space at the opposite end of the line.

**Note:** Line length counts do not account for overstriking or proportional spacing.

Lines that contain tildes (~) do not work because **ptx** uses that character internally.

## Flags

- b** *break* Uses the characters in the *break* file to separate words. Tab characters, new-line characters, and spaces are *always* used as break characters.
- f** Does not distinguish between uppercase and lowercase characters while sorting (see “**sort**” on page 672).
- g** *num* Uses *num* as the number of spaces displayed between the four parts of the line. The default *num* is 3.
- i** *ignore* Does not use any words in the *ignore* file as keywords. If the **-i** and **-o** flags are not used, **/usr/lib/eign** is the default *ignore* file.
- o** *only* Uses only the words in the *only* file as keywords.
- r** Takes any leading nonblank characters of each input line to be a reference identifier separate from the text of the line. Attaches that identifier as a fifth field on each output line.
- t** Prepares the output for the phototypesetter.
- w** *num* Uses *num* as the length of the output line. The default line length is 72 characters for **nroff** and 100 for **troff**.

## Files

/bin/sort  
/usr/lib/eign  
/usr/lib/tmac/tmac.ptx

## Related Information

The following commands: “**nroff**” on page 525 and “**troff**” on page 526.

The **mm** and **mptx** miscellaneous facilities in *AIX Operating System Technical Reference*.

The “Overview of International Character Support” in *Managing the AIX Operating System*.

## puttext

---

## puttext

---

### Purpose

Updates an output file that contains message/insert/help descriptions.

### Syntax

puttext *-n* *infile* *outfile*

OL805209

### Description

The **puttext** command uses the message/insert/help descriptions in *infile* to change, delete and add message/insert/help text to *outfile* for a component. (For information about the format and contents of *infile*, see *AIX Operating System Programming Tools and Interfaces*.)

The *infile* parameter specifies the name of the file where the message/insert/help descriptions reside. See *AIX Operating System Programming Tools and Interfaces* for a discussion of the **gettext** output file parameters that describes the format and contents of this file.

The *outfile* parameter specifies the name of the output file. If you specify an *outfile* that does not exist, a new component file is created. If you specify an existing *outfile*, a copy of that file is renamed as a backup file. In this case, an old backup file will be deleted.

**Note:** In order for the new file to be accessed by the message support run-time services, the output file name must be in the format *xxxxcc\_EN.m*. If you do not specify *outfile*, the component ID is prefixed to **\_EN.m** to form the output file name.

### Flag

- n** Causes **puttext** to assign available index numbers to the input descriptions. If you specify this flag, all the index number fields of the input file must be underscore characters or blanks.

## **Related Information**

The following commands: “**gettext**” on page 370.

The discussion of **puttext** in *AIX Operating System Programming Tools and Interfaces*.

## pwck

---

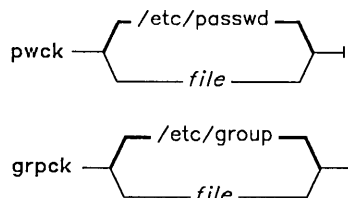
## pwck

---

### Purpose

Checks the password and group files for inconsistencies.

### Syntax



OL805008

### Description

The **pwck** command scans the named *file* or the default file **/etc/passwd** and writes to standard output any inconsistencies. The checks include validation of the number of fields, login name, user ID, group ID, and existence of the login directory and optional program name.

The **grpck** command scans the named *file* or the default file **/etc/group** and writes to standard output any inconsistencies. The checks include validation of the number of fields, group name, group ID, and whether all login names appear in the password file. **grpck** writes to standard output any group entries that do not have login names.

### Files

`/etc/group`  
`/etc/passwd`

### Related Information

The following commands: “**groups**” on page 385, “**passwd**” on page 546, and “**users**” on page 802.

The discussion of passwords in *Managing the AIX Operating System*.

## **pwd**

---

### **Purpose**

Displays the path name of the working directory.

### **Syntax**

`pwd` —

OL805210

### **Description**

The **pwd** command writes to standard output the full path name of your current directory (from the root directory). All directories are separated by a / (slash). The root directory is represented by the first /, and the last directory named is your current directory.

### **Related Information**

The following command: “**cd**” on page 121.

The **fullstat** and **ffullstat** system calls in *AIX Operating System Technical Reference*.



# qdaemon

---

## qdaemon

---

### Purpose

Schedules jobs enqueued by the **print** command.

### Syntax

`qdaemon` <sup>1</sup>

<sup>1</sup> This command is not usually entered on the command line.

OL805148

### Description

The **qdaemon** is a background process (usually started by the **rc** command file) that schedules printing jobs enqueued by **print**.

### Files

|                                   |                                           |
|-----------------------------------|-------------------------------------------|
| <code>/usr/lpd/qdir/*</code>      | Print requests.                           |
| <code>/usr/lpd/stat/*</code>      | Information on the status of the devices. |
| <code>/usr/spool/qdaemon/*</code> | Temporary copies of files to be printed.  |

### Related Information

The following commands: “**lp**” on page 459, “**piobe**” on page 557, and “**print**” on page 566.

The **qconfig** file in *AIX Operating System Technical Reference*.

---

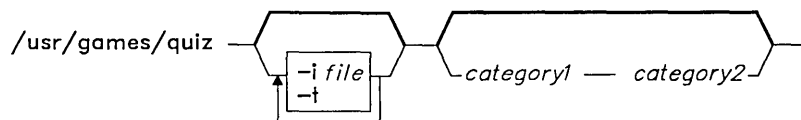
## quiz

---

### Purpose

Tests your knowledge.

### Syntax



OL805230

### Description

The **quiz** game gives associative knowledge tests on various selectable subjects. It asks about items chosen from *category1* and expects answers from *category2*. If you do not specify the categories, **quiz** gives instructions and lists the available categories.

The **quiz** game gives the correct answer whenever you press the **Enter** key by itself. The game ends when questions run out or when you press **INTERRUPT (Alt-Pause)**; **quiz** reports a score and exits.

### Flags

**-ifile** Substitutes the named *file* for the standard index file.

**Note:** In the following syntax description, brackets are normally used to indicate that an item is optional; a bold-faced bracket or brace, however, should be entered as a literal part of the syntax. A vertical list of items indicates that one and only one must be chosen. The lines in *file* must have the following syntax:

```

line      = category [ :category ] ...
category  = alternate [ |alternate ] ...
alternate = [primary]
primary   = character
           [category]
           option
option    = {category}
  
```

In an index file, the first category of each line must specify the name of an information file (the information file contains the names of files with quiz material).

## quiz

---

The remaining categories specify the order and contents of the data in each line of the information file. The quiz data in an information files follows the same syntax. A `\` (backslash) is an escape character which allows you to quote syntactically significant characters or to insert a new-line character (`\n`) into a line. When either a question or its answer is blank, **quiz** does not ask it. The construct **a!ab** does not work in an information file. Use **a{b}**.

- t Provides a tutorial. Repeats missed questions and introduce new material gradually.

## Examples

1. To start a Latin-to-English quiz:

```
/usr/games/quiz latin english
```

The **quiz** command displays Latin words and waits for you to enter what they mean in English.

2. To start an English-to-Latin quiz:

```
/usr/games/quiz english latin
```

3. To set up a Latin-English quiz, add the following line to the index file:

```
/usr/games/lib/quiz/latin:latin:english
```

This line specifies that the file `/usr/games/lib/quiz/latin` contains information about the categories `latin` and `english`.

You can add new categories to the standard index file, `/usr/games/lib/quiz/index`, or to an index file of your own. If you create your own index file, run the **quiz** command with the `-i file` flag to give it your list of quiz topics.

4. This is a sample information file:

```
cor:heart
sacerdos:priest{ess}
quando:when|since|because
optat:{{s}he |it }[desires|wishes]\
|desire|wish
alb[us|ium]:white
```

This information file contains Latin and English words. The `:` (colon) separates each Latin word from its English equivalent. Items enclosed in `{ }` (braces) are optional. A `|` (vertical bar) separates two items when entering either is correct. The `[ ]` (brackets) group items separated by vertical bars.

---

The first line accepts only the answer `heart` in response to the Latin word `cor`. The second accepts either `priest` or `priestess` in response to `sacerdos`. The third line accepts `when`, `since`, or `because` for `quando`.

The `\` (backslash) at the end of the fourth line indicates that this entry continues on the next line. In other words, the fourth and fifth lines together form one entry. This entry accepts any of the following in response to `optat`:

```
she desires    it desires    desire
she wishes    it wishes    wish
he desires     desires
he wishes     wishes
```

If you start a Latin-to-English quiz, then the last line of this sample information file instructs **quiz** to ask you the meaning of `albus`. If you start an English-to-Latin quiz, then **quiz** displays `white` and accepts `albus`, `alba`, or `album` for the answer.

If any of the characters `{`, `}`, `[`, `]`, or `!` appear in a question item, then **quiz** gives the first alternative of every `!` group and displays every optional group. Thus, the English-to-Latin question for the fourth definition in this sample is `she desires`.

## Files

```
/usr/games/lib/quiz/index
/usr/games/lib/quiz/*
```

**rc**

---

**rc**

---

## Purpose

Performs normal startup initialization.

## Syntax

`/etc/rc` —<sup>1</sup>

---

<sup>1</sup> This command is not usually run from the command line.

OL805339

## Description

When the **init** process starts up the system in normal operating mode, it runs the command file `/etc/rc` to perform the necessary system initialization, including the enabling of various loggers. If the system is being brought up with no file system checking, **init** passes the argument **m** to **rc**. If **init** determines that the root file system needs consistency checking, it passes the argument **d** to **rc**.

The contents of `/etc/rc` may be installation specific, but there are a few things that it should do:

- Run the **fsck** command to check the default file systems if **rc** is passed the **d** flag.
- Mount the default file systems (note that the root file system is implicitly mounted).
- Purge temporary files.
- Start system demons.
- Enable default ports.

If all of the necessary operations complete successfully, the file exits with a zero return code that allows **init** to start loggers to complete normal initialization and startup.

## Related Information

The following commands: “**fsck**, **dfck**” on page 333 and “**init**” on page 396.

The discussion of starting up the system in *Managing the AIX Operating System*.

---

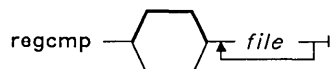
## regcmp

---

### Purpose

Compiles patterns.

### Syntax



OL805211

### Description

The **regcmp** command compiles the pattern in *file*, placing its output in *file.i*.

In most cases, **regcmp** makes unnecessary the use of the **regcmp** system call in your C programs, saving execution time and program size. The output of **regcmp** is C source code. Make each file entry a C variable name, followed by one or more blanks, followed by a pattern enclosed in double quotation marks (" "). Compiled patterns are initialized **char** declarations. Thus, *file.i* can be *included* in C programs, and *file.c* can be a file parameter to the **cc** command. The C program that uses **regcmp** output should use the **regex** subroutine to apply it to a string. (See **regcmp** and **regex** in *AIX Operating System Technical Reference*.)

### Flag

- Places the output in *file.c*

### Related Information

The **regcmp** subroutine in *AIX Operating System Technical Reference*.

# restore

---

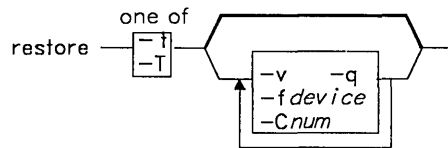
# restore

---

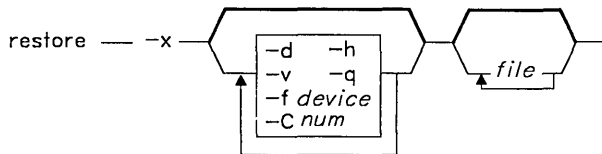
## Purpose

Copies back files created by the **backup** command.

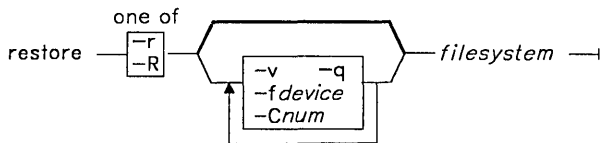
## Syntax



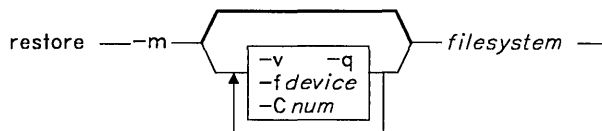
OL805251



OL805352



OL805353



OL805441

## Description

The **restore** command reads backup files written by the **backup** command and copies them back into the file system. There are four ways to use it:

- To display a table of contents for the backup (`-T`) or to display label information (`-t`)
- To restore individually named files (`-x`)

- To restore an entire file system (**-r**) or begin at an arbitrary volume number (**-R**).
- To restore an entire minidisk (**-m**).

If more than one diskette is required, **restore** reads the one mounted, then halts and requests a new one. After inserting the new diskette, press the **Enter** key to continuing restoring files.

**Note:** If the file system you are restoring is mounted and is not the root file system, **restore** unmounts the file system before it performs an i-node restore and then remounts the file system before quitting.

If the file systems you are restoring include the root file system, **restore** ensures that the other file systems are not in use. If one is, it warns you of this and quits.

## Flags

- Cnum** Specifies the number of blocks to read in a single input operation. If you do not specify this flag, **restore** selects a default value appropriate for the physical device you have selected. Larger values of *num* result in longer physical transfers from tape devices. **restore** always ignores the value of the **-C** flag when it reads a diskette; the input is always read in clusters that occupy a complete track.
- d** Indicates that if *file* is a directory, all files in that directory should be restored. In this case, the name of each restored file is always its name as shown by **restore -T**, whether the backup was by name or by i-node. The *file* names supplied need not be directories. Thus, for i-node backups:
- ```
restore -x a/b/file.c
```
- creates a file whose name is its i-node number, while:
- ```
restore -xd a/b/file.c
```
- creates a file named `a/b/file.c`. With this flag, *file* names can include pattern-matching characters, although you must quote these characters to prevent their expansion by the shell.
- Use this flag only when you are restoring by individual file name (**-x**).
- fdevice** Specifies the local input device. By default, **restore** reads the device defined in the **backupdev** entry of `/etc/filesystems` if you are restoring an entire file system and `/dev/rfd0` if you are restoring a minidisk image, restoring individually named files, or displaying a table of contents or if no **backupdev** is defined in `/etc/filesystems`. The **restore** command recognizes a special syntax for the names of input files. If the device parameter is a range of names, for example `/dev/rfd0-3`, **restore** automatically goes from one drive (in the range) to the next. After using all of the specified drives, it stops and requests that another diskette be inserted.



## restore

---

- h** Specifies that the access and modification times of restored files are to be set to the time of restoration. (When **restore** is run under superuser authority, the default action is to use the file access and modification times from the backup medium.) If a restored file is an archive, the modification times in all the member headers are also set to the time of restoration. You can specify this flag only when you are restoring individually named files.
- m** Restores an entire minidisk as an exact image.
- Note:** You can use this flag only with minidisks that are at least as large as the original minidisk that was backed up. If the minidisk is larger than the original, the leftover space becomes unusable after restoring the minidisk. You can use **restore -t** to see how large a minidisk you need.
- q** Specifies that the removable medium is ready to use. In this case, **restore** proceeds without prompting you to prepare the removable medium.
- r** Restores an entire file system. Use this flag with i-node backups only (see “**backup**” on page 76). *filesystem* can be a device name (block or character special file) or a directory name that **restore** looks up in */etc/filesystems*.
- If you are restoring a *full* (level 0) **backup**, run the **mkfs** command to create an empty file system before doing the restore. If you are restoring an *incremental backup* at, say, level 2, run **mkfs**, then restore the appropriate level 0 backup, then the level 1 backup, and finally the level 2 backup.
- Warning:** If you do not follow this procedure carefully, you can ruin an entire file system. As an added safety precaution, run **fsck** after you restore each backup level.
- R** Restarts an aborted **restore** at a specified point. **restore** prompts you for the starting volume number. This flag is invalid in combination with the **-m** flag.
- T** Displays the backup file header and the names of the backed up files. If the backup was made by name (**backup -i**), the names displayed are the ones you provided to **backup**. If the backup was made by i-node, **restore** displays the i-number of each file along with the file name. The names are relative to the root directory of the file system backed up. The only exception is the root directory itself, whose name is given as a slash (/).
- t** Displays only the backup file header.
- v** Reports the progress of the restoration as it proceeds.
- x** Restores individually named files. The names must be in the same form as the names shown by **restore -T**. With a name backup, **restore** gives the restored file whatever name was supplied when the file was backed up. If the original name was specified relative to the current directory, **restore** creates a file relative to the current directory. **restore** automatically creates any needed

directories. With an i-node backup, the name of the restored file is the same as its i-number. This flag is invalid with the **-m** flag.

## Examples

1. To list the names of files previously backed up:

```
restore -T
```

If individual files were backed up, then only the file names are displayed. If an entire file system was backed up, then the i-number is also shown.

**Note:** Unless you specify otherwise with the **-f** flag, **restore** reads the **/dev/rfd0** as the default backup device.

2. To display technical information about a backup:

```
restore -t
```

This displays information including when the backup was made, which file system was saved, and whether it is an individual file backup (*Files backed up by name*), a minidisk backup (*Files backed up by minidisk*), or a file system backup (*Files backed up by inode*).

**Note:** Files must be restored using the same method by which they were backed up.

3. To copy files to the main file system:

```
restore -x -v
```

This extracts all the files from the backup diskette and restores them to their proper places in the file system (**-x**). This displays a progress report as each file is restored (**-v**). If a file system backup is being restored, then the files are named with their i-numbers.

4. To copy selected files:

```
restore -x -v /u/jim/manual/chap1
```

This extracts the file `/u/jim/manual/chap1` from the backup diskette and restores it. To work properly, `/u/jim/manual/chap1` must be a name displayed by **restore -T**.

5. To copy all the files in a directory:

```
restore -x -d -v manual
```

This restores the directory `manual` and all the files in it. A directory named `manual` is created in the current directory to hold the files being restored.

## restore

---

6. To restore an entire file system backup:

```
mkfs /dev/hd1
restore -r -v /dev/hd1
```

This restores an entire file system backup onto /dev/hd1. It destroys and replaces any file system that was previously stored on /dev/hd1. If the backup was made using incremental file system backups, restore the backups in increasing backup-level order (0, 1, 2 . . .).

7. To restore a minidisk:

```
restore -m /dev/hd1
```

This restores the exact image of minidisk /dev/hd1. You can also identify the minidisk by its stanza name in the `/etc/filesystems` file.

## Files

|                               |                                   |
|-------------------------------|-----------------------------------|
| <code>/etc/filesystems</code> | Consulted for default parameters. |
| <code>/dev/rfd0</code>        | Default restore device.           |

## Related Information

The following command: “**backup**” on page 76.

The discussion of **filesystems** and **backup** in *AIX Operating System Technical Reference*.

“Backing up and Restoring Files” in *Using the AIX Operating System*.

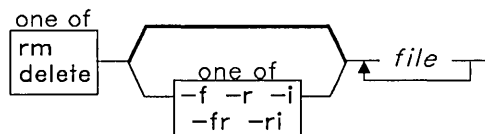
---

**rm**

---

**Purpose**

Removes files or directories.

**Syntax**

OL805212

**Description**

The **rm** (**delete**) command removes the entries for *files* from a directory. If an entry is the last link to a file, it is destroyed. To remove a file, you must have write permission in its directory, but neither read nor write permission for the file itself if you own it or are acting with superuser authority.

If a file has no write permission and standard input is a work station, **rm** displays the file permission code and reads a line from standard input. If that line begins with *y*, **rm** deletes the file; otherwise it remains.

**Flags**

- f** Does not prompt before removing a write-protected file.
- i** Prompts you before deleting each file. When you use both **-i** and **-r** together, **rm** also asks if you want to examine directories.
- r** Permits recursive removal of directories and their contents (for cases where *file* is a directory).

**Examples**

1. To delete a file:  

```
rm myfile
```

If there is another link to this file, then the file remains under that name, but the name *myfile* is removed. If *myfile* is the only link, the file itself is deleted.

2. To delete a file silently:

```
rm -f core
```

This removes **core** without asking any questions or displaying any error messages. This is normally used in shell procedures. It prevents confusing messages from being displayed when deleting files that may or may not exist.

3. To delete files one by one:

```
rm -i mydir/*
```

This interactively asks you if you want to remove each file. After each file name is displayed, enter **y** to delete the file, or press **Enter** alone to keep it.

4. To delete a directory tree:

```
rm -ir manual
```

This recursively removes the contents of all subdirectories of **manual**, then removes **manual** itself, asking if you want to remove each file. For example:

```
You: rm -ir manual
System: directory manual:
You: y
System: directory manual/draft1:
You: y
System: manual/draft1/chapter1:
You: y
System: manual/draft1/chapter2:
You: y
System: manual/draft1:
You: y
System: directory manual/draft2
You: y
System: manual/draft2:
You: n
System: manual:
You: y
```

Here, **rm** first asks if you want it to search the directory **manual**. Because **manual** contains directories, **rm** next asks for permission to search **manual/draft1** for files to delete, and then asks if you want it to delete the files **manual/draft1/chapter1** and **manual/draft1/chapter2**. **rm** next asks for permission to search the directory **manual/draft2**, and then asks for permission to delete the directories **manual/draft1**, **manual/draft2**, and **manual**. Because you denied permission to

remove manual/draft2, **rm** will not remove manual. Instead, you will see the message `rmdir: manual not empty`.

## **Related Information**

The following commands: “**del**” on page 234 and “**ln**” on page 450.

The **unlink** system call in *AIX Operating System Technical Reference*.

# rmidel

---

## rmidel

---

### Purpose

Removes a delta from a Source Code Control System (SCCS) file.

### Syntax

```
rmidel -rSID file
```

OL805213

### Description

The **rmidel** command removes the delta specified by *SID* from each named **Source Code Control System (SCCS) file**. You can remove only the most recently created delta in a branch, or the latest trunk delta if it has no branches. In addition, the *SID* you specify must not be a version currently being edited for the purpose of making a delta. To remove a delta, you must either own the SCCS file and the directory, or you must be the user who created the delta you want to remove.

If you specify a directory in place of *file*, **rmidel** performs the requested actions on all SCCS files (those with file names that have the *s.* prefix). If you specify a - (minus) in place of *file*, **rmidel** reads standard input, and interprets each line as the name of an SCCS file. **rmidel** continues to take input until it reads an end-of-file character (**Ctrl-D**).

### Flag

**-rSID** Removes the delta *SID* from the SCCS file. This flag is required.

### Related Information

The following commands: “**delta**” on page 236, “**get**” on page 359, “**help**” on page 391, and “**prs**” on page 574.

The **sccsfile** file in *AIX Operating System Technical Reference*.

The discussion of SCCS in *AIX Operating System Programming Tools and Interfaces*.

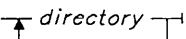
## rmdir

---

### Purpose

Removes a directory.

### Syntax

```
rmdir 
```

OL805252

### Description

The **rmdir** command removes a *directory* from the system. The *directory* must be empty before you can remove it, and you must have write permission in its parent directory. Use the **li -l** command to see if the *directory* is empty.

### Example

To empty and remove a directory:

```
rm mydir/* mydir/*.*
rmdir mydir
```

This removes the contents of *mydir*, then removes the empty directory. The **rm** command displays an error message about trying to remove the directories **.** (dot) and **..** (dot dot), and then **rmdir** removes them.

Note that **rm mydir/\* mydir/\*.\*** first removes files with names that do not begin with a dot, then those with names that do begin with a dot. You may not realize that the directory contains file names that begin with a dot because the **li** command does not normally list them.

### Related Information

The following command: “**rm**” on page 601.

The **unlink** and **rmdir** system calls in *AIX Operating System Technical Reference*.



# runacct

---

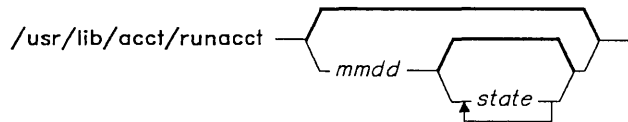
## runacct

---

### Purpose

Runs daily accounting.

### Syntax



OL805253

### Description

The **runacct** command is the main daily accounting shell procedure. Normally initiated by **cron**, **runacct** processes connect, fee, disk, queueing system and process accounting data files. It also prepares summary files for the **prdaily** procedure or for billing purposes.

The **runacct** command protects active accounting files and summary files in the event of run-time errors. It records its progress by writing descriptive messages into the file **/usr/adm/acct/nite/active**. When **runacct** encounters an error, it writes a diagnostic message to **/dev/console**, sends **mail** to users **root** and **adm**, and exits.

The **runacct** procedure also creates two temporary files, **lock** and **lock1** in the directory **/usr/adm/acct/nite**, which it uses to prevent two simultaneous calls to **runacct**. It uses the file **lastdate** (in the same directory), to prevent more than one invocation per day.

The **runacct** command breaks its processing into separate, restartable **states**. As it completes each state, it writes the name of the next state in **/usr/adm/acct/nite/statefile**. **runacct** processes the various states in the following order:

| State    | Actions                                                                               |
|----------|---------------------------------------------------------------------------------------|
| SETUP    | Moves the active accounting files to working files and restarts the active files.     |
| WTMPFIX  | Verifies the integrity of the <b>wtmp</b> file, correcting date changes if necessary. |
| CONNECT1 | Calls <b>acctcon1</b> to produce connect session records.                             |

---

|            |                                                                                                                                                                          |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CONNECT2   | Converts connect session records into total accounting records ( <b>tacct.h</b> format).                                                                                 |
| PROCESS    | Converts process accounting records into total accounting records ( <b>tacct.h</b> format).                                                                              |
| MERGE      | Merges the connect and process total accounting records.                                                                                                                 |
| FEES       | Converts the output of <b>chargefee</b> into total accounting records ( <b>tacct.h</b> format) and merges them with the connect and process total accounting records.    |
| DISK       | Merges disk accounting records with connect, process, and fee total accounting records.                                                                                  |
| QUEUEACCT  | Sorts the queue (printer) accounting records, converts them into total accounting records ( <b>tacct.h</b> format), and merges them with other total accounting records. |
| MERGETACCT | Merges the daily total accounting records in <b>daytacct</b> with the summary total accounting records in <b>/usr/adm/acct/sum/tacct</b> .                               |
| CMS        | Produces command summaries in the file <b>/usr/adm/acct/sum/cms</b> .                                                                                                    |
| USEREXIT   | If the shell file <b>/usr/adm/siteacct</b> exists, calls it at this point to perform site-dependant processing.                                                          |
| CLEANUP    | Deletes temporary files and exit.                                                                                                                                        |

To restart **runacct** after a failure, first check the **/usr/adm/acct/nite/active** file for diagnostic messages, then fix any damaged data files such as **pacct** or **wtmp**. Remove the **lock** files and **lastdate** file (all in the **/usr/adm/acct/nite** directory), before restarting **runacct**. You must specify the **mmdd** parameter if you are restarting **runacct**. It specifies the month and day for which **runacct** is to rerun the accounting. **runacct** determines the entry point for processing by reading **statefile**. To override this default action, specify the desired *state* on the **runacct** command line. For a more detailed discussion of restarting **runacct**, see *Managing the AIX Operating System*.

It is not usually a good idea to restart **runacct** in the **SETUP state**. Instead, perform the setup actions manually and restart accounting with the **WTMPFIX** state, as follows:

```
runacct mmdd WTMPFIX
```

If **runacct** fails in the **PROCESS** state, remove the last **ptacct** file, because it will be incomplete.

## Examples

1. To start **runacct**:

```
nohup /usr/lib/acct/runacct 2> /usr/adm/acct/nite/accterr &
```

## runacct

---

This starts **runacct** in the background (&), ignoring all INTERRUPT and QUIT signals (**nohup**). All standard error output is written to the file `/usr/adm/acct/nite/accterr`.

2. To restart **runacct**:

```
nohup /usr/lib/acct/runacct 0601 2>> /usr/adm/acct/nite/accterr &
```

This restarts **runacct** for the day of June 1 (0601). **runacct** reads the file `/usr/adm/acct/nite/statefile` to find out the state to begin with. Standard error output is added to the end of the file `/usr/adm/acct/nite/accterr`.

3. To restart **runacct** in a specific state, in this case the **MERGE** state:

```
nohup /usr/lib/acct/runacct 0601 MERGE 2>> /usr/adm/acct/nite/accterr &
```

## Files

|                                             |                                                     |
|---------------------------------------------|-----------------------------------------------------|
| <code>/usr/adm/wtmp</code>                  | Login/logout history file.                          |
| <code>/usr/adm/pacct*</code>                | Process accounting file.                            |
| <code>/usr/adm/acct/nite/daytacct</code>    | Disk usage accounting file.                         |
| <code>/usr/adm/qacct</code>                 | Active queue accounting file.                       |
| <code>/usr/adm/fee</code>                   | Record of fees charge to users.                     |
| <code>/usr/adm/acct/sum/*</code>            | Command and total accounting summary files.         |
| <code>/usr/adm/acct/nite/ptacct*.mmd</code> | Concatenated version of <b>pacct</b> files.         |
| <code>/usr/adm/acct/nite/active</code>      | <b>runacct</b> message file.                        |
| <code>/usr/adm/acct/nite/lock*</code>       | Prevent simultaneous invocation of <b>runacct</b> . |
| <code>/usr/adm/acct/nite/lastdate</code>    | Contains last date <b>runacct</b> was run.          |
| <code>/usr/adm/acct/nite/statefile</code>   | Contains current state to process.                  |

## Related Information

The following commands: “**acct/\***” on page 31, “**acctcms**” on page 36, “**acctcom**” on page 38, “**acctcon**” on page 42, “**acctmerg**” on page 46, “**acctprc**” on page 48, “**cron**” on page 172, and “**fwtmp**” on page 345.

The **acct** system call and the **acct** and **utmp** files in *AIX Operating System Technical Reference*.

“Running System Accounting” in *IBM RT PC Managing the AIX Operating System*.


## sact

---

### Purpose

Displays current Source Code Control System (SCCS) file editing status.

### Syntax

sact 

OL805063

### Description

The **sact.** command reads *Source Code Control System* (SCCS) files and writes to standard output the contents, if any, for the *p-file* associated with *file* (see “SCCS Files” on page 360 for information on the contents of the p-file). If - (minus) is specified for *file*, **sact.** reads standard input, and interprets each line as the name of an SCCS file. If *file* is a directory, **sact.** performs its actions on all SCCS files (that is, those files with the *s.* prefix).

### Related Information

The following commands: “**delta**” on page 236, “**get**” on page 359, and “**unget**” on page 790.

The **sccsfile** file in *AIX Operating System Technical Reference*.

The discussion of SCCS in *AIX Operating System Programming Tools and Interfaces*.

**sadc****sadc****Purpose**

Provides a system activity report package.

**Syntax**

```

/usr/lib/sa/sadc { interval — num } { outfile }
/usr/lib/sa/sa1 { interval — num }
/usr/lib/sa/sa2 1

```

<sup>1</sup> See the **sar** command for the format and flag description. Note that you cannot use the **-o** and **-f** flags with **sa2**.

OL805254

**Description**

The operating system contains a number of counters that are incremented as various system actions occur. They include the following:

- System unit utilization counters
- Buffer usage counters
- Disk and tape I/O activity counters
- TTY device activity counters
- Switching and system-call counters
- File-access counters
- Queue activity counters
- Inter-process communications counters

The **sadc** command and the **sa1** and **sa2** shell procedures sample, save and process this data.

**Note:** These commands only report on local activities.

## sadc

The **sadc** command, the data collector, samples system data *num* times every *interval* seconds. It writes in binary format to *outfile* or to the standard output. If you do not specify *interval* or *num*, a special record is written. This facility is used at system startup to mark the time when the counter restarts from zero.

## sa1

Use the shell procedure **sa1**, a variant of **sadc** to collect and store binary data in the file */usr/adm/sa/sadd*, where *dd* is the day of the month. The *interval* and *num* parameters specify that the record should be written *num* times at *interval* seconds. If you do not specify these parameters, one record is written. You must have permission to write in the directory */usr/adm/sa* to use this command.

The **sa1** command is designed to be started automatically by the **cron** command.

## sa2

Use the shell procedure **sa2**, a variant of the **sar** command, to write a daily report in the file */usr/adm/sa/sar $dd$* . See “**sar**” on page 614 for a description of the flags.

The **sa2** command is designed to be started automatically by the **cron** command.

## Files

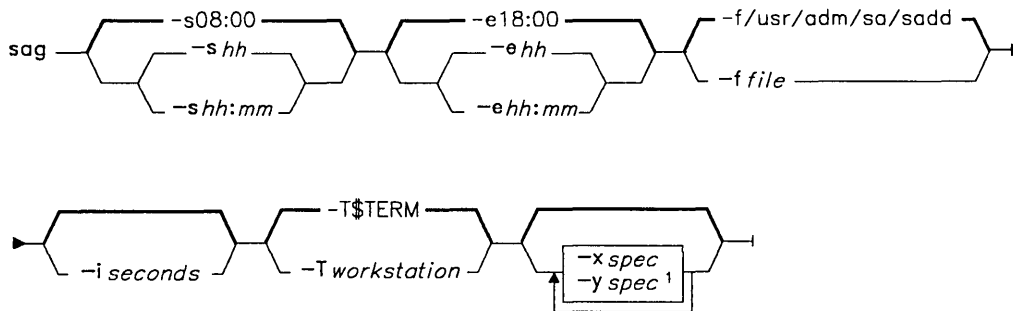
|                                       |                                                               |
|---------------------------------------|---------------------------------------------------------------|
| <i>/usr/adm/sa/sadd</i>               | Daily data file, <i>dd</i> represents the day of the month.   |
| <i>/usr/adm/sa/sar<math>dd</math></i> | Daily report file, <i>dd</i> represents the day of the month. |
| <i>/tmp/sa.adrfl</i>                  | Address file.                                                 |

## Related Information

The following commands: “**cron**” on page 172, “**sag**” on page 612, “**sar**” on page 614, and “**timex**” on page 755.

**sag****sag****Purpose**

Displays a graph of system activity.

**Syntax**

<sup>1</sup> The default for `-y` is `'% usr 0 100; % usr + % sys 0 100; % usr + % sys + %wio 0 100'`

OL805387

**Description**

The **sag** command displays a graph of system activity. It gets information either from the daily activity file `usr/adm/sa/sadd` or from the binary data file selected by the `-f` flag. You must have already created this file by running the **sar** command with the `-o` flag. (See “**sar**” on page 614.)

The **sag** command calls the **sar** command, selecting the desired data by string-matching the data column header.

**Flags**

The **sag** command passes the first four of the following flags to **sar** in order to collect the desired data for display. The last three flags specify plotting parameters.

`-e hh[:mm]` Selects data up to the time specified by `hh[:mm]`). The default time is 18:00.

- 
- f** *file* Reads data from *file*. The default *file* is `/usr/adm/sa/sadd`, the current daily data file.
  - i** *seconds* Selects data at intervals as close as possible to *seconds*.
  - s** *hh[:mm]* Selects data later than the specified time. Default is 08:00.
  - T** *workstation* Produces output suitable for *workstation*. (See “**tplot**” on page 762 for known work stations.) If you do not specify a work station, **sag** uses the value found in the shell variable **\$TERM**.
  - x** *spec* Specifies the x axis. *spec* has the following form:

*name* [*opname*] . . . [*lo hi*]

where *name* is a character string matching a column header in the **sar**-created data file (with an optional device name in brackets), or it is an integer value. *op* is +, -, \*, or / surrounded by blanks, with up to five *names* specified. Parentheses are not recognized and evaluation is left to right. Note that + and - have precedence over \* and / in evaluating expressions. *lo* and *hi* specify numeric scale limits. If these limits are unspecified, **sag** gets these limits from the data.

- y** *spec* Specifies the y axis. *spec* has the same form as **x spec**.

Specify only one *spec* for the x axis. If unspecified, the x axis assumes the time specified with the **-e** and **-s** flags (or their defaults if they are not used) as x axis limits. You can specify up to five *specs* separated by : (semicolons) for **-y**. If unspecified, the y axis has the value:

**-y** "%usr 0 100; %usr + %sys 0 100; %usr + %sys + %wio 0 100"

If you include blanks or an escaped carriage return (**\Enter**) within the **-x** and **-y specs**, enclose them in " " (double quotation marks).

## Files

`/usr/adm/sa/sadd` Daily data file for day *dd*.

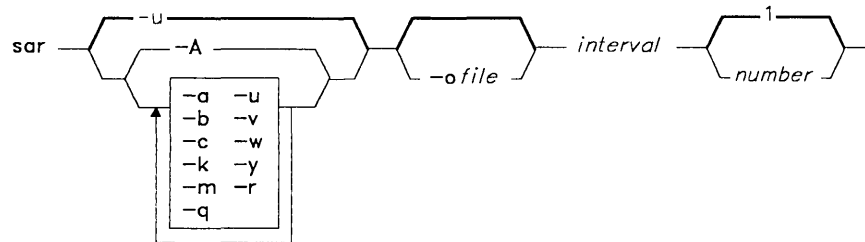
## Related Information

The following commands: “**sar**” on page 614 and “**tplot**” on page 762.

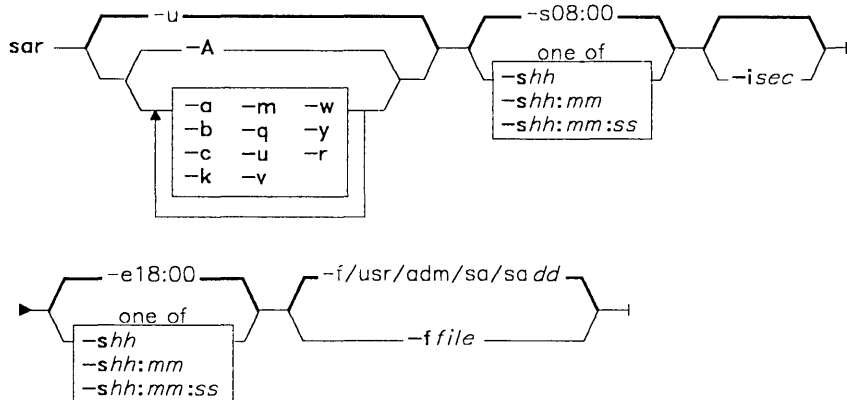


**sar****sar****Purpose**

Collects, reports, or saves system activity information.

**Syntax**

OL805390



OL805369

**Description**

The first format of the `sar` command writes to standard output the contents of selected cumulative activity counters in the operating system. It writes information a total of *number* times spaced *interval* seconds apart. The default value of *number* is 1. You can also save the collected data in the file specified by `-o file`.

In the second format (with no sampling interval specified), **sar** extracts and writes to standard output records previously saved in a file. This file can be either the one specified by the **-f** flag or, by default, the standard system activity daily data file, **/usr/adm/sa/sadd**, for the current day, *dd*.

You can select with flags the system activity you want information about. Not specifying any flags selects only **cpu** activity. Specifying the **-A** flag selects all activities.

**Note:** This command only reports on local activities.

## Flags

- a** Reports use of file access system routines:
  - iget/s** Calls per second to the i-node look-up routine.
  - namei/s** Calls per second to the directory search routine.
  - dirblk/s** Directory blocks read per second by namei().
- A** Report all data.
- b** Reports buffer activity for transfers, accesses, and cache hit ratios:
  - lread/s, lwrit/s** Number of logical read/write requests per interval.
  - bread/s, bwrit/s** Number of block read/write operations per interval.
  - %rcache, %wcache** Cache hit ratios (for example, 1 - bread/lread).
  - pread/s, pwrit/s** Read/writes per interval on seekable raw devices.
- c** Reports system calls:
  - scall/s** Total number of system calls per second.
  - rchar/s, wchar/s** Characters transferred per interval by read/write calls.
  - sread/s, swrit/s**
  - fork/s, exec/s** Specific system calls per second.
- e hh[:mm][:ss]]** Sets the ending time of the report. The default ending time is 18:00.
- f file** Extracts records from *file* (created by **-o file**). The default *file* is the current daily data file, **/usr/adm/sa/sadd**.
- i seconds** Selects data records at intervals as close as possible to the specified number of *seconds*. Otherwise, **sar** reports all intervals found in the data file.
- k** Reports kernel activity:
  - ksched/s** Number of kernel processes assigned to tasks per second.
  - kproc-ov/s** Number of overflows occurring between sampling points.
  - kexit/s** Number of kernel processes terminating per second.
- m** Reports message and semaphore activities:
  - msg/s** IPC message primitives per second.
  - sema/s** IPC semaphore primitives per second.

## sar

---

- o** *file* Saves the readings in *file* in binary form. Each reading is in a separate record and each record contains a tag identifying the time of the reading.
- q** Reports average queue length while occupied, and percentage of time occupied:  
runq-sz, %runocc Runs queue of processes in memory and runnable.
- r** Reports VRM paging statistics:  
slots The number of free pages on the paging minidisk.  
cycle/s The number of page replacement cycles per second.  
fault/s The number of page faults per second.  
odio/s The number of nonpaging disk I/Os per second.
- s** *hh[:mm[:ss]]* Sets the starting time of the data. That is, extract records time-tagged at or following the time specified. The default starting time is 08:00.
- u** Reports CPU activity (this flag is on by default):  
%usr Percentage of CPU time devoted to the user.  
%sys Percentage of CPU time devoted to the kernel.  
%wio Percentage of CPU time waiting for block I/O to complete.  
%idle Percentage of CPU time idle.
- v** Reports status of text, process, i-node, and file tables:  
text-sz, proc-sz, inod-sz, file-sz Entries in use at each sample point for each table.  
text-ov, proc-ov, inod-ov, file-ov Overflows occurring at each sample point for each table.
- w** Reports system switching activity:  
pswch/s Process switches per second.
- y** Reports TTY device activity:  
rawch/s Tty raw input queue characters per second.  
canch/s TTY canonical input queue characters per second.  
outch/s TTY output queue characters per second.  
revin/s TTY receive interrupts per second.  
xmtin/s TTY transmit interrupts per second.  
mdmin/s TTY modem interrupts per second.

## Files

`/usr/adm/sa/sadd` Daily data file, where *dd* are numbers representing the day of the month.

## Related Information

The following command: “**sag**” on page 612.

The discussion of monitoring system activity in *Managing the AIX Operating System*.

# sccsdiff

---

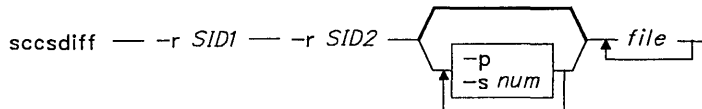
## sccsdiff

---

### Purpose

Compares two versions of a Source Code Control System (SCCS) file.

### Syntax



OL805258

### Description

The `sccsdiff` command reads two versions of an **SCCS** file, compares them, and writes to standard output the differences between the two versions. Any number of **SCCS** files can be specified, but the same arguments apply to all files.

### Flags

- `-p` Pipes the output through `pr`.
- `-rSID1` Specifies `SID1` as one delta of the **SCCS** file for `sccsdiff` to compare.
- `-rSID2` Specifies `SID2` as the other delta of the **SCCS** file for `sccsdiff` to compare.
- `-snum` Specifies the file segment size for `bdiff` to pass to `diff`. This is useful when `diff` fails due to a high system load.

### Related Information

The following commands: “`bdiff`” on page 88, “`get`” on page 359, “`help`” on page 391, and “`pr`” on page 561.

The `sccsfile` file in *AIX Operating System Technical Reference*.

The discussion of **SCCS** in *AIX Operating System Programming Tools and Interfaces*.

---

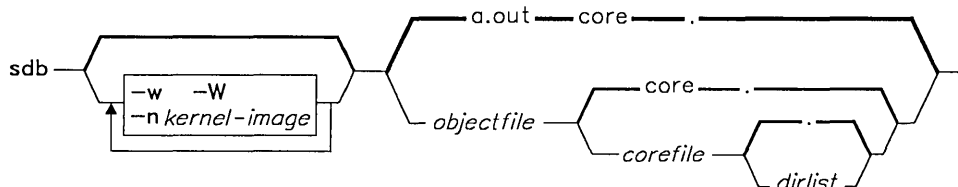
## sdb

---

### Purpose

Provides a symbolic debugger for C and assembler programs.

### Syntax



OL805214

### Description

The **sdb** command provides a symbolic debugger to be used with C and assembler programs. With it you can examine object and core files and provide a controlled environment for running a program. You can set breakpoints at selected statements or run the program one line at a time. You can debug using symbolic variables and instruct **sdb** to display them in their correct format.

Normally, *objectfile* is an executable file produced by invoking **cc** with the **-g** flag. If you have not compiled *objectfile* using the **-g** flag or if it is not executable (because of compiler or loader errors), the symbolic capabilities of **sdb** are limited, but you can still examine the file and debug the program. *objectfile* should always be in the same directory as the source files used to construct it. Its default name is **a.out**.

The *corefile* parameter specifies a core image file. Its default is **core**. The system writes out this core image of a process when it ends abnormally. Specifying **-** (minus) for *corefile* instructs **sdb** to ignore any core image file that may be present. The colon-separated list of directories specified by the *dirlist* parameter identifies the location of the source files used to build *corefile*. The default is the current directory. If *dirlist* is the name of a file, the contents of the file should be a colon-separated list of directory names.

While running, **sdb** always recognizes a *current line* and *current file*. If *corefile* exists, **sdb** initially sets the current line and the current file to the line and the file that contains the source statement at which the process ended. Otherwise, it sets them to the first line in **main** and the file containing **main**. There is also a *current function*, which is the function you are working with at any given time. You can change the current line, file, or function with the **e** command.

Write variable names as you do in C language programs. Access variables local to a function by using the form *function:variable*. The current function is the default function. You can also specify a variable by its address. Since you can use all forms of integer constants which are valid in C, addresses can be expressed as decimal, octal, or hexadecimal values.

Refer to structure members as *variable.member*, pointers to structure members as *variable->member*, and array elements as *variable[number]*.

If you use the form *number.member* or *number->member*, **sdb** assumes *number* to be the address of the last structure referred to. Generally, **sdb** interprets a structure as a set of variables. Thus, it displays the values of all elements when you request it to display a structure. If, however, you request the address of a structure, it displays this value and not the addresses of individual elements.

Refer to elements of a multidimensional array as *variable[number][number]* or as *variable[number,number]*. In place of *number*, use the form *number;number* to indicate a range of values. You can also use an \* (asterisk) to represent all legitimate values for a subscript or omit subscripts to indicate the full range of values. As with structures, **sdb** displays all the values of an array or of a section of an array if you omit trailing subscripts. If you omit subscripts, it displays only the address of the array itself or of the section specified.

Refer to a variable on the stack by using the form *function:variable,*. Here, *number* specifies the variable's location on the stack, counting the top, or most recently pushed variable, as the first. Use this for recursive function calls. The current function is the default.

Refer to line numbers as *filename:number* or *function:number*. The current file and current function are the default values.

**Note:** Data stored in text sections is indistinguishable from functions.

Line number information in optimized functions is unreliable, and some information may be missing.

Source line and local symbol information for routines in shared libraries is not implemented, and these modules should not be compiled with the **-g** flag. Break points may be set in these routines by address only, and code in shared library modules may be single-stepped by instruction only.

The **sdb** command cannot comprehend a module in which C functions (as opposed to declarations and preprocessor definitions) occur in include files.

## Flags

- n** *kernel-image* Specifies the name of the running kernel (or the one running when *corefile* was produced). This enables proper traces back through the floating-point emulation code. **/unix** is the default value.
- w** Allows overwriting of locations in *objectfile*.
- W** Turns off the warnings normally given if source files cannot be found or are newer than *objectfile*.

## Subcommands

### Examining Program Data

- T** Displays the top line of the stack trace.
- t** Displays a stack trace of the program that ended abnormally.
- variable/[nlf]*** Displays *variable* or *n* memory locations starting at the address of *variable*. The *l* parameter selects the number of bytes in one memory location. Your choices are:
  - b** One byte
  - h** Two bytes
  - l** Four bytes.
 The *f* parameter selects the display format. This can be one of the following:
  - a** Displays all bytes from the address of the *variable* to the first null byte.
  - c** Displays a character value.
  - d** Displays a decimal value.
  - f** Displays a 32-bit, single-precision floating-point value.
  - g** Displays a 64-bit, double-precision floating-point value.
  - I** Interprets values as assembly language instructions and displays numerically.
  - i** Interprets values as assembly language instructions and displays them numerically and symbolically.
  - o** Displays an octal value.
  - t** Displays F if *variable* = 0; otherwise, displays T.
  - p** Displays a pointer to a function.
  - s** Treats *variable* as a string pointer and displays characters beginning with the address to which it points and ending at the first null byte reached.
  - u** Displays an unsigned decimal value.



**x** Displays a hexadecimal value.

If you do not specify *n*, *l*, or *f*, **sdb** chooses a value appropriate to *variable* type as declared in the source file. Specifying a memory location size only works with formats **c**, **d**, **o**, **u**, and **x**. You can specify the number of memory locations (*n*) to be displayed by the **s** or **a** formats. For strings that contain two-byte extended characters, the font shift character is represented by a backslash followed by lowercase **s** and the font shift number. For example, `\s1` means that the current byte being displayed is a font shift character. This form of representation for the font shift byte is only available when a count is specified. However, if the first character contained in the address specified by the **a** format is the second byte of a two-byte extended character, then that byte is displayed without the proper shift affixed to construct the whole two-byte sequence. The default action for these formats is to display characters until either a null byte is reached or 128 characters have been displayed. The command `./` (dot slash) redisplay the last variable.

You can use the special **sh** characters `*` (asterisk) and `?` (question mark) in function and variable names, providing a limited form of pattern matching. If you give no function name, global variables and variables local to the current function are matched. If you specify a function name, then only variables local to that function are matched. To match only global variables, use the form `:pattern`.

The **sdb** command recognizes structures, arrays, and pointers so that all of the following commands work:

```
array[2][3]/  
sym.id/  
psym->usage/  
xsym[20].p->usage/
```

`line?[lf]`  
`variable?[lf]`

Displays the value found in *objectfile* at the address selected by *line* or *variable* (function name), using the specified *length* and *format*. The default format is **i**.

`line = [lf]`  
`variable = [lf]`  
`number = [lf]`

Displays the address of *variable* or *line* or the value of *number* in the specified length and format. The default is **lx**. `number = [lf]` provides a convenient way to convert decimal, octal, and hexadecimal values.

`variable!value`

Sets *variable* to the given *value*. *value* may be a numeric or character constant or another variable. Expressions that produce more than one value, such as structures, are not allowed as *value*. However, *variable* may be an expression which represents more than one variable, such as an array or structure name.

Specify a character constant with an initial ' (single quote), for example, 'c'. Numbers are treated as integers unless they contain a decimal point or an exponent. In the latter case, they are treated as having the type double. Register values are viewed as integers. If you give an address of a variable, it is treated as the address of a variable of type **int**. C conventions are used in any type conversions that are necessary to perform the indicated assignment.

- x** Displays the machine registers and the current assembly language instruction.
- X** Displays the current assembly language instruction.

## Displaying and Manipulating Source Files

*e function*  
*e file*  
*e dir/*  
*e dir file*

Changes the current function, file, or directory. Specifying only *function* also sets the current file to the one containing the selected function. **sdb** reports the current function, file, or directory for any unspecified parameters.

*/pattern/*

Searches forward from the current line for a line containing a string matching *pattern*. The trailing / (slash) can be omitted. See “**ed**” on page 280 for a discussion of pattern notation.

*?pattern?*

Searches backward from the current line for a line containing a string matching *pattern*. The trailing ? (question mark) can be omitted.

**p**

Displays the current line.

**z**

Displays the current line and the following nine lines. Sets the current line to the last line displayed.

**w**

Displays the 10 lines around the current line (a window).

*number*

Sets the current line to *number*. Displays the new current line.

*number+*

Advances the current line by the specified *number* of lines. Displays the new current line.

*number-*

Decreases the current line by the specified *number* of lines. Displays the new current line.

**Ctrl-D**

Scrolls. Pressing **Ctrl-D** displays the next 10 lines of source or data.

**Enter**

If the previous command displayed a source line, pressing the **Enter** key advances the current line by one line and displays the new current line. If the previous command displayed a memory location, pressing the **Enter** key displays the next memory location.

## Controlling the Running of the Source Program

**[num] r** [*p* [*p2*] . . . ]

**[num] R** Runs the program with the given parameters. If you specify no parameters with **r**, it reuses previously specified parameters. **R** runs the program with no parameters. A parameter beginning with **<** (left angle bracket) or **>** (right angle bracket) redirects input or output, respectively. If given, *num* selects the number of breakpoints to be ignored.

**[line] b** [*command*; *command*] . . . ]

Sets a breakpoint at the given line. If you specify a function name without a line number, **sdb** places a breakpoint at the first line in the function, even if it was not compiled with the **-g** flag. If you do not specify a *line*, a breakpoint is placed at the current line. If you specify no *commands*, the program stops running just before the breakpoint and returns control to **sdb**. Otherwise **sdb** performs the specified *commands* when the breakpoint is encountered, and then the program being debugged continues. If the **k** command is specified, however, control returns to **sdb**.

**B** Lists the currently active breakpoints.

**[line] d** Deletes a breakpoint at the selected line. If you select no *line*, breakpoints are deleted interactively. **sdb** displays each breakpoint location and reads a line from standard input. If the line begins with a **y** or **d**, then it deletes the breakpoint.

**D** Deletes all breakpoints.

**[line] c** [*num*]

**[line] C** [*num*]

Continues running program after a breakpoint or an interrupt. **C** continues after resetting the signal that caused the program to stop. **c** ignores the signal. An optional *num* selects the number of breakpoints to ignore. If you specify a *line*, **sdb** places a temporary breakpoint at the line and continues the program. It deletes the breakpoint when the command finishes.

**[line] g** [*num*] Continues after a breakpoint, with execution resumed at the given line. *num* specifies how many breakpoints to ignore.

**l** Displays the last executed line.

**i**

**I**

Runs the program one machine level instruction at a time, ignoring the signal that stopped the program (**i**) or passing the signal back to the program (**I**).

- 
- s** [*num*]  
**S** [*num*]      Runs the program for one or the specified number of lines. **S** is equivalent to **s** except that it does not stop within called functions. Use **S** if you are confident that the called function works, but want to test the calling routine.
- variable***\$m** [*num*]  
**address:m** [*num*]      Runs the program until the specified location is modified with a new value or is modified a specified *num* of times. The *variable* must be accessible from the current function.
- line a*      If *line* is of the form *function:number*, this command has the effect of the **sdb** subcommand: *line b l*. If *line* is of the form *function:*, it has the effect of the **sdb** subcommand: *function: b l*.
- [*level*] **v**      Toggles verbose mode, for use with the **S**, **s** or **m** commands. If you omit *level*, then just the current source file or subroutine name is displayed when either changes. If *level* is 1 or greater, each C source line is displayed before it is executed; if *level* is 2 or greater, each assembler statement is also displayed. If verbose mode is on for any level, another **v** turns it off.
- function*(*p [p . . . ]*)[*f*]      Runs the named function, passing to it the specified parameters. These can be integer, character, or string constants or the names of variables accessible to the current function. You can specify the format of displayed values. The default format is **d** (decimal).
- k**      Kills the program being debugged.
- M**      Displays the address maps. Program addresses are mapped to file addresses using two field triples: *b1, e1, f1* and *b2, e2, f2*. The *f1* field is the length of the header at the beginning of the file; the *f2* field is the displacement from the beginning of the file to the data. For a plain executable file with mixed text and data, this is the same as the length of the header; for shared text and split instruction/data files, this is the length of the header plus the size of the text portion.
- The *b* and *e* fields are the starting and ending locations for a segment. Given an address *A*, calculate its location in the file (either **a.out** or **core**) as follows:
- If  $b1 < A < e1$   
then *file address* =  $(A-b1) f1$
- If  $b2 < A < e2$   
then *file address* =  $(A-b2) f2$

## sdb

---

- M[?/][\*] b e f** Records new values for the address map. The parameters **?** and **/** specify the text and data maps respectively. The first segment is changed unless you specify **\***, in which case the second segment is changed. (These segments differ only in programs with split instruction and data space. In this case, use the second segment to examine the data section of the **a.out** file rather than the data section of the core image file.) If you give fewer than three values, the remaining map parameters are left unchanged.
- " string** Displays the given string. **sdb** recognizes C escape sequences of the form *\character*.
- !AIX-command** Performs the specified *AIX-command*.
- <file** Reads commands from *file* until reaching the end of file and then continues to accept commands from standard input. When a command in such a file tells **sdb** to display a variable, the variable name is displayed along with the value. This command cannot be nested.
- >file** Redirects standard output to *file*.
- q** Exits the debugger.

### Debugging the Debugger

- Q** Displays a list of functions and files being debugged.
- V** Displays the version number.

## Files

a.out, core

## Related Information

The following commands: “**cc**” on page 112, “**ed**” on page 280, and “**sh**” on page 637.

The **a.out** and **core** files in *AIX Operating System Technical Reference*.

The “Overview of International Character Support” in *Managing the AIX Operating System*.

The topic “Debugging Programs” in *AIX Operating System Programming Tools and Interfaces*.

---

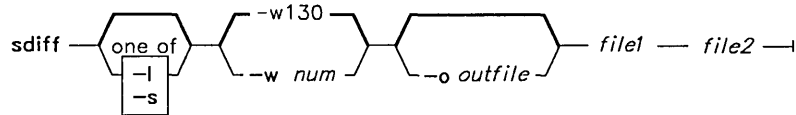
# sdiff

---

## Purpose

Compares two files and displays the differences in a side-by-side format.

## Syntax



OL805301

## Description

The **sdiff** command reads *file1* and *file2*, uses **diff** to compare them, and writes the results to standard output in a side-by-side format. **sdiff** displays each line of the two files with a series of blanks between them if the lines are identical, a < (less than sign) in the field of blanks if the line only exists in *file1*, a > (greater than sign) if the line only exists in *file2*, and a | (vertical bar) for lines that are different.

When you specify the **-o** flag, **sdiff** produces a third file by merging *file1* and *file2* according to your instructions.

## Flags

- l** Displays only the left side when lines are identical.
- o outfile** Creates a third file, *outfile*, by a controlled line-by-line merging of *file1* and *file2*. The following subcommands govern the creation of this file:
  - l** Adds the left side to *outfile*.
  - r** Adds the right side to *outfile*.
  - s** Stops displaying identical lines.
  - v** Begins displaying identical lines.

## sdiff

---

- e l**
- e r**
- e b**
- e** Starts **ed** with the left side, the right side, both sides, or an empty file, respectively.  
  
Each time you exit from **ed**, **sdiff** writes the resulting edited file to the end of *outfile*. If you fail to save the changes before exiting, **sdiff** writes the initial input to *outfile*.
- q** Exits the program.
- s** Does not display identical lines.
- w num** Sets the width of the output line to *num*, 130 characters, by default.

## Examples

1. To print a comparison of two files:  

```
sdiff chap1.bak chap1 | print
```

This prints a side-by-side listing that compares each line of *chap1.bak* and *chap1*. The `| print` sends the listing to the **print** command. **sdiff** assumes that your printer has wide paper (130 columns).
2. To display only the lines that differ:  

```
sdiff -s -w 80 chap1.bak chap1
```

This displays the differences at the work station. The **-w 80** sets page width to 80 columns. The **-s** flag tells **sdiff** not to display lines that are identical in both files.
3. To selectively combine parts of two files:  

```
sdiff -s -w 80 -o chap1.combo chap1.bak chap1
```

This combines *chap1.bak* and *chap1* into a new file called *chap1.combo*. For each group of differing lines, **sdiff** asks you which group to keep or whether you want to edit them using **ed**.

## Related Information

The following commands: “**diff**” on page 246 and “**ed**” on page 280.

---

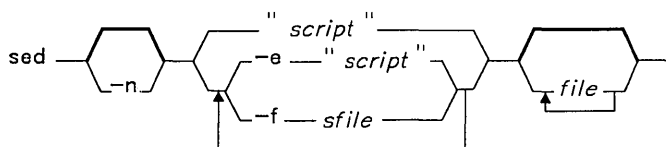
# sed

---

## Purpose

Provides a stream editor.

## Syntax



OL805302

## Description

The **sed** command modifies lines from the specified *file* according to an edit script and writes them to standard output. The **sed** command includes many features for selecting lines to be modified and making changes only to the selected lines.

The **sed** command uses two work spaces for holding the line being modified: the **pattern space**, where the selected line is held, and the **hold space**, where a line can be stored temporarily.

An edit script consists of individual subcommands, each one on a separate line. The general form of **sed** subcommands is:

```
[address-range] function[modifiers]
```

The **sed** command processes each input *file* by reading an input line into a pattern space, applying all **sed** subcommands in sequence whose addresses select that line, and writing the pattern space to standard output. It then clears the pattern space and repeats this process for each line in the input *file*. Some of the subcommands use a hold space to save all or part of the pattern space for subsequent retrieval.

When a command includes an address, either a line number or a search pattern, only the addressed line or lines is affected by the command. Otherwise, the command is applied to all lines.

An address is either a decimal line number, a \$ (dollar sign), which addresses the last line of input, or a context address. A context address is a regular expression similar to those used in **ed** except for the following differences:



- You can select the character delimiter for patterns. The general form of the expression is:

`\?pattern?`

where `?` is a character delimiter you select. This delimiter cannot be a two-byte international character support extended character. The default form for the pattern is:

`/pattern/`

- The sequence `\n` matches a new-line character in the pattern space, except the terminating new line.
- A `.` (dot) matches any character except a terminating new-line character. That is, unlike `ed`, which cannot match a new-line character in the middle of a line, `sed` can match a new-line character in the pattern space.

Certain commands allow you to specify one line or a range of lines to which the command should be applied. These commands are called addressed commands. The following rules apply to addressed commands:

- A command line with no address selects every line.
- A command line with one address, expressed in context form, selects each line that matches the address.
- A command line with two addresses separated by commas selects the entire range from the first line that matches the first address through the next line that matches the second. (If the second address is a number less than or equal to the line number first selected, only one line is selected.) Thereafter the process is repeated, looking again for the first address.

**Note:** The *text* parameter accompanying the `a\`, `c\`, and `i\` commands can continue onto more than one line provided all lines but the last end with a `\` to quote the new-line character. Backslashes in text are treated like backslashes in the replacement string of an `s` command and can be used to protect initial blanks and tabs against the stripping that is done on every script line. The *rfile* and *wfile* parameters must end the command line and must be preceded by exactly one blank. Each *wfile* is created before processing begins.

## Flags

- `-e "script"` Uses the file *script* as the editing script. If you are using just one `-e` flag and no `-f` flag, the `-e` flag may be omitted.
- `-f sfile` Uses *sfile* as the source of the edit script. *sfile* is a prepared set of editing commands to be applied to *file*.
- `-n` Suppresses all information normally written to standard output.

---

## Subcommands

In the following list of functions, the maximum number of permissible addresses for each function is indicated in parentheses. The **sed** script subcommands are as follows:

- (1) **a**\  
*text* Places *text* on the output before reading the next input line.
- (2) **b**[*label*] Branches to the : command bearing the *label*. If *label* is empty, it branches to the end of the script.
- (2) **c**\  
*text* Deletes the pattern space. With 0 or 1 address or at the end of a 2-address range, places *text* on the output. Starts the next cycle.
- (2) **d** Deletes the pattern space. Starts the next cycle.
- (2) **D** Deletes the initial segment of the pattern space through the first new-line character. Starts the next cycle.
- (2) **g** Replaces the contents of the pattern space by the contents of the hold space.
- (2) **G** Appends the contents of the hold space to the pattern space.
- (2) **h** Replaces the contents of the hold space by the contents of the pattern space.
- (2) **H** Appends the contents of the pattern space to the hold space.
- (1) **i**\  
*text* Writes *text* to standard output before reading the next line into the pattern space.
- (2) **l** Writes the pattern space to standard output showing nondisplayable characters as two-digit octal values. Long lines are folded.
- (2) **n** Writes the pattern space to standard output. Replaces the pattern space with the next line of input.
- (2) **N** Appends the next line of input to the pattern space with an embedded new-line character. (The current line number changes.) You can use this to search for patterns that may be split onto two lines.
- (2) **p** Writes the pattern space to standard output.
- (2) **P** Writes the initial segment of the pattern space through the first new-line character to standard output.
- (1) **q** Branches to the end of the script. Does not start a new cycle.
- (2) **r** *rfile* Reads the contents of *rfile*. Places contents on the output before reading the next input line.

## sed

---

- (2)*s/pattern/replacement/flags*  
Substitutes the *replacement* string for the first occurrence of the *pattern* in the pattern space. Any character appearing after the *s* can substitute for the */* separator.
- You can add zero or more of the following *flags*:
- g** Substitutes all nonoverlapping instances of the *pattern* rather than just the first one.
  - p** Writes the pattern space to standard out if a replacement was made.
  - w *wfile***  
Writes the pattern space to *wfile* if a replacement was made. Appends the pattern space to *wfile*. If *wfile* was not already created by a previous write by this **sed** script, **sed** creates it.
- (2)*tlabel* Branches to *:label* in the script file if any substitutions were made since the most recent reading of an input line execution of a **t** subcommand. If you do not specify *label*, control transfers to the end of the script.
- (2)*wwfile* Appends the pattern space to *wfile*.
- (2)**x** Exchanges the contents of the pattern space and the hold space.
- (2)*y/pattern1/pattern2/*  
Replaces all occurrences of characters in *pattern1* with the corresponding characters *pattern2*. The byte lengths of *pattern1* and *pattern2* must be equal.
- (2)**!sed-cmd** Applies the specified **sed** subcommand only to lines *not* selected by the address or addresses.
- (0):*label* This script entry simply marks a branch point to be referenced by the **b** and **t** commands. This label can be any sequence of eight or fewer bytes.
- (1)=  
Writes the current line number to standard output as a line.
- (2){*subcmd*  
.  
:  
.  
.  
}  
Groups subcommands enclosed in {} (braces).

---

## Examples

1. To perform a global change:

```
sed "s/happy/enchanted/g" chap1 >chap1.new
```

This replaces each occurrence of `happy` found in the file `chap1` with `enchanted`, and puts the edited version in a separate file named `chap1.new`. The `g` at the end of the `s` subcommand tells `sed` to make as many substitutions as possible on each line. Without the `g`, `sed` replaces only the first `happy` on a line.

The `sed` stream editor operates as a filter. It reads text from standard input or from the files named on the command line (`chap1` in this example), modifies this text, and writes it to standard output. Unlike most editors, it does not replace the original file. This makes `sed` a powerful command when used in pipelines.

2. To use `sed` as a filter in a pipeline:

```
pr chap2 | sed "s/Page *[0-9]*$/(&)/" | print
```

This encloses the page numbers in parentheses before printing `chap2`. The `pr` command puts a heading and page number at the top of each page, then `sed` puts the page numbers in parentheses, and the `print` command prints the edited listing.

The `sed` pattern `/Page *[0-9]*$/` matches page numbers that appear at the end of a line. The `s` subcommand changes this to `(&)`, where the `&` (ampersand) stands for the page number that was matched.

3. To display selected lines of a file:

```
sed -n "/food/p" chap3
```

This displays each line in `chap3` that contains the word `food`. Normally, `sed` copies every line to standard output after it is edited. The `-n` flag stops `sed` from doing this. You then use subcommands like `p` to write specific parts of the text. Without the `-n`, this example would display *all* the lines in `chap3`, and it would show each line containing `food` twice.

4. To perform complex editing:

```
sed -f script.sed chap4 >chap4.new
```

It is always a good idea to create a `sed` script file when you want to do anything very complex. You can then test and modify your script before using it. You can also reuse your script to edit other files. Create the script file with an interactive text editor.

5. A sample **sed** script file:

```
:join
/\\$/ {N
s/\\n//
b join
}
```

This **sed** script joins each line that ends with a `\` (backslash) to the line that follows it.

First, the pattern `/\\$/` selects a line that ends with a `\` for the group of commands enclosed in `{ }`. The `N` subcommand then appends the next line, imbedding a new-line character. The `s/\\n//` deletes the `\` and imbedded new-line character. Finally, `b join` branches back to the label `:join` to check for a `\` at the end of the newly joined line. Without the branch, **sed** writes the joined line and read the next one before checking for a second `\`.

**Note:** The `N` subcommand causes **sed** to stop immediately if there are no more lines of input (that is, if `N` reads the end-of-file character). It does not copy the pattern space to standard output before stopping. This means that if the last line of the input ends with a `\`, then it is not copied to the output.

## Related Information

The following commands: “**awk**” on page 70, “**ed**” on page 280, and “**grep**” on page 381.

## | **setdma**

---

### | **Purpose**

| Sets the DMA channel adapter of the specified adapter.

### | **Syntax**

| `setdma` one of  
eesdi 0  
eesdi 1 |

OL805466

### | **Description**

| The **setdma** command sets the DMA channel of the specified adapter. **eesdi 0** sets the  
| DMA adapter channel to 0. **eesdi 1** sets the DMA adapter channel to 1.

### | **Related Information**

| *Installing and Customizing the AIX Operating System.*

**setdma**

---

---

## setmnt

---

### Purpose

Creates mount table.

### Syntax

```
setmnt —l
```

OL805062

### Description

The **setmnt** command reads lines from standard input and writes the **/etc/mnttab** table to standard output (see the **mnttab** file in *AIX Operating System Technical Reference*). The **/etc/mnttab** is needed for both the **mount** and **unmount** commands. **setmnt** creates a **mnttab** entry for each line read. Input lines have the format:

```
filesys directory
```

where *filesys* is the name of the file system's special file and *directory* is the root name of that file system. Thus, *filesys* and *directory* become the first two strings in the **mnttab** entry. *filesys* and *directory* must not be longer than 100 characters.

The **setmnt** command enforces an upper limit on the maximum number of **mnttab** entries.

### Examples

To set the mount table after it has been destroyed or made invalid:

```
setmnt <<end
/dev/hd1 /u
/dev/fd0 /a
/dev/fd1 /b
end
```

This sets the mount table to show **/dev/hd1** mounted on **/u**, **/dev/fd0** on **/a**, and **/dev/fd1** on **/b**.

The **<<end** and **end** define a Here Document, which uses the text entered before the **end** line as the standard input for the **setmnt** command. For more details, see "Inline Input Documents" on page 650.



## setmnt

---

### Files

*/etc/mnttab*

### Related Information

The **mnttab** file in *AIX Operating System Technical Reference*.

---

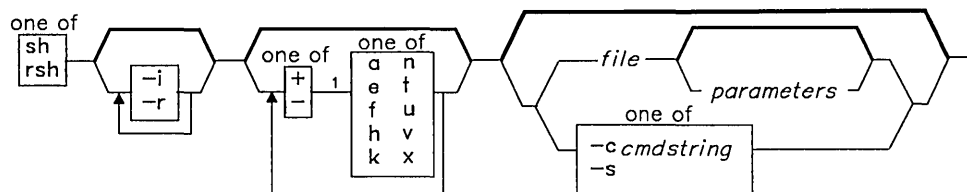
# sh

---

## Purpose

Interprets commands read from a file or entered at the keyboard.

## Syntax



OL805425

<sup>1</sup> Do not put a blank between these items.

## Description

The **sh** command is a system command interpreter and programming language. It is not a part of the operating system *kernel*, but an ordinary user program that reads commands entered at the keyboard and arranges for their execution. In addition, it can read commands that you have saved in a file. Such a file is usually called a *shell procedure* or a *command file*. For a complete description of how to write shell procedures to take advantage of this useful tool, see *Using the AIX Operating System*.

A restricted version of shell (the **rsh** command) is available that allows you to create user environments with a limited set of privileges and capabilities. See “Restricted Shell” on page 658 for additional information on the restricted shell.

## Commands

A *simple command* is a sequence of *words* separated by blanks or tabs. A word is a sequence of characters and/or numerals that contains no unquoted blanks. The first word in the sequence (numbered as 0), usually specifies the name of a command. Any remaining words, with a few exceptions, are passed to that command.

The *value* of a simple command is its exit value if it ends normally or (octal) 200 *status* if it ends abnormally. For a list of *status* values, see the **signal** system call in *AIX Operating System Technical Reference*.

A **command** is either a simple command or a **control command** (see “Control Commands” on page 653).

A **pipeline** is a sequence of one or more commands separated by a | (vertical bar) or, for historical compatibility, by a ^ (circumflex). In a pipeline, the standard output of each command becomes the standard input of the next command. Each command runs as a separate process, and the shell waits for the last command to end. A **filter** is a command that reads its standard input, transforms it in some way, then writes it to its standard output. A pipeline normally consists of a series of filters. Although the processes in a pipeline (except the first process) can execute in parallel, they are synchronized to the extent that each program needs to read the output of its predecessor.

The exit value of a pipeline is the exit value of the last command.

A **list** is a sequence of one or more pipelines separated by ; (semicolon), & (ampersand), && (two ampersands), or || (two vertical bars) and optionally ended by a ; (semicolon) or an & (ampersand). These separators and terminators have the following effects:

- ;  
Causes **sequential execution** of the preceding pipeline (the shell waits for the pipeline to finish).
- &  
Causes **asynchronous execution** of the preceding pipeline (the shell does **not** wait for the pipeline to finish).
- &&  
Causes the list following it to be executed **only** if the preceding pipeline returns a zero exit value.
- ||  
Causes the list following it to be executed **only** if the preceding pipeline returns a nonzero exit value.

**Note:** The **cd** command is an exception. If it returns a nonzero exit value, no subsequent commands in a list are executed, regardless of the separator characters.

The ; and & separators have equal precedence, as do && and ||. The single-character separators have lower precedence than the double-character separators. An unquoted new-line character following a pipeline functions the same as a ; (semicolon).

The shell treats as a comment any word that begins with a # character and ignores that word and all characters following up to the next new-line character.

## Command Execution

Each time the shell executes a command, it carries out the substitutions discussed in the following text. If the command name matches one of the built-in commands discussed in “Built-in Commands” on page 654, it executes it in the shell process. If the command name does not match a built-in command but matches the name of a defined function, it executes the function in the shell process. The shell sets the positional parameters to the parameters of the function.

If the command name matches neither a built-in command nor the name of a defined function and the command names an executable file that is a compiled (binary) program,

---

the shell (as *parent*) spawns a new (*child*) process that immediately runs the program. If the file is marked executable but is not a compiled program, the shell assumes that it is a shell procedure. In this case, the shell spawns another instance of itself (a *subshell*), to read the file and execute the commands included in it (note how this differs from the execution of functions). The shell also executes a parenthesized command in a subshell (see page 654). From your point of view as a user, a compiled program is run in exactly the same way as a shell procedure.

The shell normally searches for commands in four places in the file system. The shell first looks for the command in the `/bin` directory. If it does not find the command there, it looks in the `/usr/bin` directory. If this also fails, it looks in the `/etc` directory and then, finally, in the current directory. You can also give a specific path name when you invoke a command, for example `/bin/sort`, in which case the shell does not search any directories other than the one you specify in the path name. If the command name contains a `/` (slash), the shell does not use the search path (note that the restricted shell will not execute such commands). You can give a full path name that begins with the root directory (as in `/bin/sort`), or a path name relative to the current directory, for example `bin/myfile`. In this last case, the shell looks in the current directory for a directory named `bin` and in that directory for `myfile`.

You can change the particular sequence of directories searched by resetting the `PATH` variable (page 646).

The shell remembers the location in the search path of each executed command (to avoid unnecessary `execs` later). If the command was found in a *relative directory* (one whose name does not begin with `/`), the shell must redetermine its location whenever the current directory changes. The shell forgets all remembered locations whenever you change the `PATH` variable or execute the `hash -r` command (page 655).

## Signals

The shell ignores `INTERRUPT` and `QUIT` signals for an invoked command if the command is terminated with a `&` (that is, if it is running in the background). Otherwise signals have the values inherited by the shell from its parent, with the exception of signal 11 (see also the built-in `trap` command on page 657).

## The .profile File

When you log in, the shell is called to read your commands. Before it does that, however, it checks to see if a file named `/etc/profile` exists on the system, and if it does, it reads commands from it (this file should set variables needed by all users). After this, the shell looks for a file named `.profile` in your login directory. If it finds one, it executes commands from it. Finally, the shell is ready to read commands from your standard input.

## File-name Substitution

Command parameters are very often file names. You can automatically produce a list of file names as parameters on a command line by specifying a pattern that the shell matches against the file names in a directory.

Most characters in such a pattern match themselves, but you can also use some special pattern-matching characters in your pattern. These special characters are:

- \* Matches any string, including the null string.
- ? Matches any one character.
- [ . . . ] Matches any one of the characters enclosed in square brackets.
- [! . . . ] Matches any character *other than* one of the characters that follow the exclamation mark within square brackets.

Inside square brackets, a pair of characters separated by a - (minus) specifies a set of all characters lexically within the inclusive range of that pair, according to the current collating sequence (see “**ctab**” on page 204). The **NLCTAB** environment variable controls the collating sequence.

The current collating sequence may group characters into *equivalence classes* for the purpose of defining the end points of a range of characters. For example, if the collating sequence defines the lexical order to be `AaBbCc . . .` and groups upper- and lowercase characters into equivalence classes, then all the following have the same effect: `[a-c]`, `[A-C]`, `[a-C]`, and `[A-c]`.

Pattern matching has some restrictions. If the first character of a file name is a . (dot), it can be matched only by a pattern that literally begins with a dot. For example, `*` matches the file names `myfile` and `yourfile` but not the file names `.myfile` and `.yourfile`. To match these files, use a pattern such as the following:

```
.*file
```

If a pattern does not match any file names, then the pattern itself is returned as the result of the attempted match.

File and directory names should not contain the characters `*`, `?`, `[`, or `]` because this can cause infinite recursion (that is, infinite loops) during pattern-matching attempts.

---

## Shell Variables and Command-Line Substitutions

The shell has several mechanisms for creating variables (assigning a string value to a name). Certain variables, *positional parameters* and *keyword parameters*, are normally set only on a command line. Other variables are simply names to which you or the shell can assign string values.

### *Positional Parameters*

When you run a shell procedure, the shell implicitly creates positional parameters that reference each word on the command line by its position on the command line. The word in position 0 (the procedure name), is called \$0, the next word (the first parameter) is called \$1, and so on up to \$9. To refer to command line parameters numbered higher than 9, use the built-in **shift** command (page 656).

You can also assign values to these positional parameters explicitly by using the built-in **set** command (page 656).

### *User-defined Variables*

The shell also recognizes alphanumeric variables to which string values can be assigned. You assign a string value to a *name*, as follows:

```
name=string
```

A name is a sequence of letters, digits, and underscores that begins with an underscore or a letter. To use the value that you have assigned to a variable, add a \$. (dollar sign) to the beginning of its name. Thus \$*name* yields the value *string*. Note that no blanks surround the = (equal sign) in an assignment statement. (Positional parameters cannot appear in an assignment statement; they can only be set as described earlier.) You can put more than one assignment on a command line, but remember: the shell performs the assignments from right to left.

If you surround *string* with quotation marks, either double or single (" " ' '), the shell does not treat blanks, tabs, semicolons, and new-line characters within it as word delimiters but imbeds them literally in the string.

If you surround *string* with double quotation marks (" "), the shell still recognizes variable names in the string and performs *variable substitution*; that is, it replaces references to positional parameters and other variable names that are prefaced by \$ with their corresponding values, if any. The shell also performs *command substitution* (see "Command Substitution" on page 647) within strings that are surrounded by double quotation marks.

If you surround *string* with single quotation marks ( ' '), the shell does no variable or command substitution within the string. The following sequence illustrates this difference:

```
You: stars=*****
      asterisks1="Add $stars"
      asterisks2='Add $stars'
      echo $asterisks1
Display: Add *****
You: echo $asterisks2
Display: Add $stars
```

The shell does not reinterpret blanks in assignments after variable substitution (see “Blank Interpretation” on page 653). Thus the following assignments result in `$first` and `$second` having the same value:

```
first='a string with embedded blanks'
second=$first
```

When you reference a variable, you can enclose the variable name (or the digit designating a positional parameter) in `{ }` (braces) to delimit the variable name from any following string. In particular, if the character immediately following the name is a letter, digit, or underscore and the variable is not a positional parameter, then the braces are required:

```
You: a='This is a'
      echo "${a}n example"
Display: This is an example
You: echo "$a test"
Display: This is a test
```

See “Conditional Substitution” on page 643 for a different use of braces in variable substitutions.

### ***A Command's Environment***

All the variables (with their associated values) that are known to a command at the beginning of its execution constitute its *environment*. This environment includes variables that a command inherits from its parent process and variables specified as keyword parameters on the command line that calls the command.

The shell passes to its child processes the variables that have been named as arguments to the built-in **export** command. **export** places the named variables in the environments of both the shell and all its future child processes.

Keyword parameters are variable-value pairs that appear in the form of assignments, normally before the procedure name on a command line (but see also the **-k** flag on page 656). Such variables are placed in the environment of the procedure being called.

---

For example, given the following simple procedure that echoes the values of two variables (saved in a command file named `key_command`):

```
# key_command
echo $a $b
```

the following command lines produce the output shown:

```
You: a=key1 b=key2 key_command
Display: key1 key2
You: a=tom b=john key_command
Display: tom john
```

A procedure's keyword parameters are not included in the parameter count stored in `$#`.

A procedure can access the values of any variables in its environment; however, if it changes any of these values, these changes are not reflected in the shell environment. They are local to the procedure in question. To place these changes in the environment that the procedure passes to its child processes, you must export these values within that procedure.

To obtain a list of variables that have been made exportable from the current shell, enter:

```
export
```

(You will also get a list of variables that have been made **readonly**.) To get a list of name-value pairs in the current environment, enter:

```
env
```

## ***Conditional Substitution***

Normally, the shell replaces `$variable` with the string value assigned to `variable`, if there is one. However, there is a special notation that allows ***conditional substitution***, depending on whether the variable is set and/or not null. By definition, a variable is ***set*** if it has ever been assigned a value. The value of a variable can be the null string, which you can assign to a variable in any one of the following ways:

```
A=
bcd=""
Efg=' '
set ' ' ""
```

The first three of these examples assign the null string to each of the corresponding variable names. The last example sets the first and second positional parameters to the null string and unsets all other positional parameters.



The following is a list of the available expressions you can use to perform conditional substitution:

- `${variable-string}` If the variable is set, substitute the value of *variable* in place of this expression. Otherwise, replace this expression with the value of *string*.
- `${variable:-string}` If the variable is set and is not null, substitute the value of *variable* in place of this expression. Otherwise, replace this expression with the value of *string*.
- `${variable=string}` If the variable is set, substitute the value of *variable* in place of this expression. Otherwise, set *variable* to *string* and then substitute the value of the *variable* in place of this expression. You cannot assign values to positional parameters in this fashion.
- `${variable:=string}` If the variable is set and is not null, substitute the value of *variable* in place of this expression. Otherwise, set *variable* to *string* and then substitute the value of the *variable* in place of this expression. You cannot assign values to positional parameters in this fashion.
- `${variable?string}` If the variable is set, substitute the value of *variable* in place of this expression. Otherwise, display a message of the form:  
*variable: string*  
and exit from the current shell (unless the shell is the login shell). If you do not specify *string*, the shell displays the following message:  
*variable: parameter null or not set*
- `${variable?:string}` If the variable is set and not null, substitute the value of *variable* in place of this expression. Otherwise, display a message of the form:  
*variable: string*  
and exit from the current shell (unless the shell is the login shell). If you do not specify *string*, the shell displays the following message:  
*variable: parameter null or not set*
- `${variable+string}` If the variable is set, substitute the value of *string* in place of this expression. Otherwise, substitute the null string.
- `${variable:+string}` If the variable is set and not null, substitute the value of *string* in place of this expression. Otherwise, substitute the null string.

---

In conditional substitution, the shell does not evaluate *string* until it uses it as a substituted string, so that, in the following example, the shell executes the **pwd** command only if **d** is not set or is null:

```
echo ${d:-`pwd`}
```

### *Variables Used by the Shell*

The shell uses the following variables. The shell sets some of them, and you can set or reset all of them:

|                  |                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>CDPATH</b>    | The search path for the <b>cd</b> (change directory) command (see the <b>PATH</b> variable in the following list for an explanation of search paths).                                                                                                                                                                                                                                                                                         |
| <b>HOME</b>      | The name of your <i>login directory</i> , the directory that becomes the current directory upon completion of a login. The <b>login</b> program initializes this variable. The <b>cd</b> command uses the value of <b>\$HOME</b> as its default value. If you use this variable in your shell procedures rather than using explicit full path name, your procedures run even if your login directory is changed or if another user runs them. |
| <b>LOGNAME</b>   | Your login name, marked <b>readonly</b> in the <b>/etc/profile</b> file.                                                                                                                                                                                                                                                                                                                                                                      |
| <b>MAIL</b>      | The path name of the file used by the mail system to detect the arrival of new mail. If <b>MAIL</b> is set, the shell periodically checks the modification time of this file and displays the value of <b>\$MAILMSG</b> if this time changes.<br><br>You should set <b>MAIL</b> in your <b>.profile</b> file. The value normally assigned to it by users of the <b>mail</b> command is <b>/usr/mail/\$LOGNAME</b> .                           |
| <b>MAILCHECK</b> | The number of seconds that the shell lets elapse before checking again for the arrival of mail in the files specified by the <b>MAILPATH</b> or <b>MAIL</b> parameters. The default value is 600 seconds (10 minutes). If you set <b>MAILCHECK</b> to 0, the shell checks before each prompt.                                                                                                                                                 |
| <b>MAILPATH</b>  | A colon-separated list of file names (see <b>PATH</b> ). If you set this parameter, the shell informs you of the arrival of mail in any of the files specified in the list. You can follow each file name by a <b>%</b> (percent sign) and a message to be displayed when mail arrives. Otherwise, the shell uses the value of <b>MAILMSG</b> or by default "you have mail".                                                                  |
| <b>MAILMSG</b>   | The mail notification message. If you explicitly set <b>MAILMSG</b> to a null string ( <b>MAIL=""</b> ), no message is displayed.                                                                                                                                                                                                                                                                                                             |
| <b>NLCTAB</b>    | Defines the collating sequence to use when sorting names and when character ranges occur in patterns. If absent, it may be taken from the parameter <b>NLFILE</b> . If both are absent, the American English collating                                                                                                                                                                                                                        |

- sequence is used. See “Overview of International Character Support” in *IBM RT PC Managing the AIX Operating System* for further information.
- PATH** An ordered list of directory path names separated by colons. The shell searches these directories in the specified order when it looks for commands. A null string anywhere in the list represents the current directory.
- PATH** is normally initialized in the `/etc/profile` file, usually to `/bin:/usr/bin:/etc::`. You can reset this variable to suit your own needs. Thus if you wish to search your current directory first, rather than last, you would enter:
- ```
PATH=./bin:/usr/bin:/etc
```
- where, by definition, a null string is assumed in front of the leading colon. If you have a personal directory of commands (say, `$HOME/bin`) that you want searched before the standard system directories, set your **PATH** as follows:
- ```
PATH=$HOME/bin:/bin:/usr/bin:/etc::
```
- The best place to set your **PATH** to something other than the default value is in your `.profile` file (see “The `.profile` File” on page 639). You cannot reset **PATH** if you are executing commands under the restricted shell.
- PS1** The string to be used as the primary system prompt. An interactive shell displays this prompt string when it expects input. The default value of **PS1** is “\$” (a \$ followed by a blank).
- PS2** The value of the secondary prompt string. If the shell expects more input when it encounters a new-line character in its input, it prompts with the value of **PS2**. The default value of **PS2** is “> ” (a > followed by a blank).
- IFS** The characters that are *internal field separators* (the characters that the shell uses during blank interpretation, see “Blank Interpretation” on page 653). The shell initially sets **IFS** to include the blank, tab, and new-line characters.
- SHACCT** The name of a file (that you own). If this parameter is set, the shell writes an accounting record in the file for each shell procedure executed. You can use accounting programs such as `acctcom` and `acctcms` to analyze the data collected.
- SHELL** A path name whose simple part (the part after the last /) contains an “r” if you want the shell to become restricted when invoked. This should be set and exported by the `$HOME/.profile` file of each restricted login.

---

**TIMEOUT** A number of minutes. After the shell displays its prompt, you have **TIMEOUT** minutes to enter a command. If you fail to do so, the shell exits; in the login shell, such an exit is a logout. Setting **TIMEOUT** to 0 inhibits automatic logout.

### *Predefined Special Variables*

Several variables have special meanings; the following are set *only* by the shell:

- \$# The number of positional parameters passed to the shell, not counting the name of the shell procedure itself. \$# thus yields the number of the highest-numbered positional parameter that is set. One of the primary uses of this variable is to check for the presence of the required number of arguments.
- \$? The exit value of the last command executed. Its value is a decimal string. Most UNIX commands return 0 to indicate successful completion. The shell itself returns the current value of \$? as its exit value.
- \$\$ The process number of the current process. Because process numbers are unique among all existing processes, this string of up to five digits is often used to generate unique names for temporary files. The following example illustrates the recommended practice of creating temporary files in a directory used only for that purpose:

```
temp=$HOME/temp/$$
ls >$temp
.
.
.
rm $temp
```
- #! The process number of the last process run in the background (using the & terminator). Again, this is a string of up to five digits.
- \$- A string consisting of the names of the execution flags (page 656) currently set in the shell.

### *Command Substitution*

To capture the output of any command as an argument to another command, place that command line within grave accents ( ` ` ). This concept is known as command substitution. The shell first executes the command or commands enclosed within the grave accents, and then replaces the whole expression, grave accents and all, with their output. This feature is often used in assignment statements:

```
today=`date`
```

This statement assigns the string representing the current date to the variable `today`. The following assignment saves, in the variable `files`, the number of files in the current directory:

```
files=`ls | wc -l`
```

You can enclose any command that writes to standard output in grave accents. You can nest command substitutions as long as you quote the inside sets of grave accents with a preceding `\` (backslash):

```
logmsg=`echo Your login directory is `pwd```
```

You can also give values to shell variables indirectly by using the built-in **read** command. **read** takes a line from standard input (usually your keyboard), and assigns consecutive words on that line to any variables named:

```
read first init last
```

will take an input line of the form:

```
J. Q. Public
```

and have the same effect as if you had typed:

```
first=J.   init=Q.   last=Public
```

**read** assigns any excess words to the last variable.

## Quoting Mechanisms

Many characters have a special meaning to the shell; sometimes you want to conceal that meaning. Single (`'` `'`) and double (`"` `"`) quotation marks surrounding a string or a backslash (`\`) before a single character provide this function in somewhat different ways.

Within single quotation marks, all characters (except the single quotation character itself), are taken literally, with any special meaning removed. Thus:

```
stuff='echo $? $*; ls * | wc'
```

results only in the literal string `echo $? $*; ls * | wc` being assigned to the variable `stuff`; the `echo`, `ls`, and `wc` commands are not executed, nor are the variables `?` and `*` and the special character `*` expanded by the shell.

Within double quotation marks, the special meaning of certain characters (the `$`, ```, and `"`) does persist, while all other characters are taken literally. Thus, within double quotation marks, command and variable substitution takes place. In addition, the quotation marks do not affect the commands within a command substitution that is part of the quoted string, so characters there retain their special meanings.

Consider the following sequence:

```
You:  ls *
Display: file1
        file2
        file3
You:  message="This directory contains `ls * ` "
        echo $message
Display: This directory contains file1 file2 file3
```

This shows that the `*` special character inside the command substitution was expanded.

To hide the special meaning of `$`, ```, and `"` within double quotation marks, precede these characters with a `\` (backslash). Outside of double quotation marks, preceding a character with `\` is equivalent to placing it within single quotation marks. Hence, a `\` immediately preceding the new-line character (that is, a `\` at the end of the line) hides the new-line character and allows you to continue the command line on the next physical line.

## Redirection of Input and Output

In general, most commands do not know or care whether their input or output is associated with the keyboard, the display screen, or a file. Thus a command can be used conveniently either at the keyboard or in a pipeline.

### *Standard Input and Standard Output*

When a command begins running, it usually expects that three files are already open: *standard input*, *standard output*, and *diagnostic output* (sometimes called *error output* or *standard error output*). A number called a *file descriptor* is associated with each of these files as follows:

File descriptor 0 Standard input

File descriptor 1 Standard output

File descriptor 2 Diagnostic (error) output

A child process normally inherits these files from its parent; all three files are initially assigned to the work station (0 to the keyboard, 1 and 2 to the display). The shell permits them to be redirected elsewhere before control is passed to a command. Any argument to the shell in the form `<file` or `>file` opens the specified file as the standard input or output, respectively. In the case of output, this process destroys the previous contents of *file*, if it already exists. An argument in the form `>>file` directs the standard output to the end of *file*, thus allowing you to add data to it without destroying its existing contents. If *file* does not exist, the shell creates it.

Such redirection arguments are subject only to variable and command substitution; neither blank interpretation nor pattern matching of file names occurs after these substitutions.

Thus:

```
echo 'this is a test' > *.ggg
```

produces a one-line file named `*.ggg` (a disastrous name for a file), and:

```
cat < ?
```

produces an error message, unless you have a file named `?` (also a bad choice for a file name).

### ***Diagnostic and Other Output***

Diagnostic output from UNIX commands is normally directed to the file associated with file descriptor 2. You can redirect this error output to a file by immediately preceding either output redirection arrow (`>` `>>`) with a 2 (the number of the file descriptor). For example, the following line adds error messages from the `cc` command to the file `ERRORS`:

```
cc testfile.c 2>> ERRORS
```

Note that there must be no blanks between the file descriptor and the redirection symbol; otherwise, the shell interprets the number as a separate argument to the command.

You can also use this method to redirect the output associated with any of the first 10 file descriptors (numbered 0 through 9) so that, for instance, if a command (`cmd`) writes to file descriptor 9 (although this is not a recommended programming habit), you can capture that output in a file `savedata` as follows:

```
cmd 9> savedata
```

If a command writes to more than one output, you can independently redirect each one. Suppose that a command directs its standard output to file descriptor 1, directs its error output to file descriptor 2, and builds a data file on file descriptor 9. The following command line redirects each of these outputs to a different file:

```
cmd > standard 2> error 9> data
```

### ***Inline Input Documents***

Upon seeing a command line of the form:

```
cmd << eofstring
```

where *eofstring* is any string that does not contain any pattern-matching characters, the shell takes the subsequent lines as the standard input of `cmd` until it reads a line consisting of only *eofstring* (possibly preceded by one or more tab characters). The lines

---

between the first *eofstring* and the second are frequently referred to as a *here document*. If a - (minus) immediately follows the <<, the shell strips leading tab characters from each line of the input document before it passes the line to the command.

The shell creates a temporary file containing the input document and performs variable and command substitution on its contents before passing it to the command. It performs pattern matching on file names that are a part of command lines in command substitutions. If you want to prohibit all substitutions, quote any character of *eofstring*:

```
cmd << \eofstring
```

The here document is especially useful for a small amount of input data that is more conveniently placed in the shell procedure rather than kept in a separate file (such as editor "scripts"). For instance, you could enter:

```
cat <<- xyz
    This message will be shown on the
    display with leading tabs removed.
xyz
```

This feature is most useful in shell procedures. Note that inline input documents can *not* appear within grave accents (command substitution).

### ***Input and Output Redirection Using File Descriptors***

As discussed previously, a command occasionally directs output to some file associated with a file descriptor other than 1 or 2. The shell also provides a mechanism for creating an output file associated with a particular file descriptor. By entering:

```
fd1>&fd2
```

where *fd1* and *fd2* are valid file descriptors, you can direct the output that would normally be associated with file descriptor *fd1* to the file associated with *fd2*. The default value for *fd1* and *fd2* is 1 (standard output). If, at execution time, no file is associated with *fd2*, then the redirection is void. The most common use of this mechanism is to **direct** standard error output to the same file as standard output, as follows:

```
cmd 2>&1
```

If you want to **redirect** both standard output and standard error output to the same file, enter:

```
cmd > file 2>&1
```

The order here is significant. First, the shell associates file descriptor 1 with `file`; then it associates file descriptor 2 with the file that is currently associated with file descriptor 1. If you reverse the order of the redirections, standard error output will go to the display and standard output would go to `file` because at the time of the error output redirection, file descriptor 1 was still associated with the display.



You can also use this mechanism to redirect standard input. You could enter:

```
fda<&fdb
```

to cause both file descriptors to be associated with the same input file. For commands that run sequentially, the default value of *fda* and *fdb* is 0 (standard input). For commands that run asynchronously (commands terminated by &), the default value of *fda* and *fdb* is **/dev/null**. Such input redirection is useful for commands that use two or more input sources.

### ***Summary of Redirection Options***

The following can appear anywhere in a simple command or can precede or follow a command, but they are not passed to the command:

- |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| < <i>file</i>         | Use <i>file</i> as standard input.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| > <i>file</i>         | Use <i>file</i> as standard output. Create the file if it does not exist; otherwise truncate it to zero length.                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| > > <i>file</i>       | Use <i>file</i> as standard output. Create the file if it does not exist; otherwise add the output to the end of the file.                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| < < [-] <i>eofstr</i> | Read as standard input all lines from <i>eofstr</i> up to a line containing only <i>eofstr</i> or up to an end-of-file character. If any character in <i>eofstr</i> is quoted, the shell does not expand or interpret any characters in the input lines; otherwise, it performs variable and command substitution and ignores a quoted new-line character (\new-line). Use a \ to quote characters within <i>eofstr</i> or within the input lines.<br><br>If you add a - (minus) to < < then all leading tabs are stripped from <i>eofstr</i> and from the input lines. |
| < & <i>digit</i>      | Associate standard input with file descriptor <i>digit</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| > & <i>digit</i>      | Associate standard output with file descriptor <i>digit</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| < &-                  | Close standard input.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| > &-                  | Close standard output.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

The restricted shell does not allow the redirection of output.

## Blank Interpretation

After the shell performs variable and command substitution, it scans the results for internal field separators (those defined in the shell variable **IFS**, see page 646). It splits the line into distinct words at each place it finds one of these characters. It retains explicit null arguments (" " ' ') and discards implicit null arguments (those resulting from parameters that have no values).

## Control Commands

The shell provides several flow control commands that are useful in creating shell procedures:

**for** *name* [ **in** *word* . . . ]

**do** *list*

**done**

For each *word*, sets *name* to *word* and executes the commands in *list*. If you omit **in** *word* . . ., then the **for** command executes *list* for each positional parameter that is set. Execution ends when there are no more words in the word list.

**case** *word* **in**

*pattern* [*!pattern*] . . . ) *list*;;

[ .

.

*pattern* [*!pattern*] . . . ) *list*;;]

**esac**

Executes the commands in the *list* associated with the first *pattern* that matches *word*. Uses the same character-matching notation in patterns that you use for file-name substitution (see “File-name Substitution” on page 640), except that you do not need to match explicitly a slash, a leading dot, or a dot immediately following a slash.

**if** *list*

**then** *list*

[**elif** *list*] . . .

[**else** *list*]

**fi**

Executes the *list* following the **if** keyword. If it returns a zero exit value, execute the *list* following the first **then**. Otherwise, execute the *list* following **elif** (if there is an **elif**), and if its exit value is zero, execute the next **then**. Failing that, execute the *list* following the **else**. If no **else** *list* or **then** *list* is executed, the **if** command returns a zero exit value.

**while** *list*

**do** *list*

**done**

Executes the *list* following the **while**. If the exit value of the last command in the list is zero, executes the *list* following **do**. Continue looping through the *lists* until the exit value of the last command in the **while** *list* is nonzero. If no commands in the **do** *list* are executed, the **while** command returns a zero exit value.

- until** *list*  
**do** *list*  
**done** Executes the *list* following the **until**. If the exit value of the last command in the list is nonzero, executes the *list* following **do**. Continues looping through the *lists* until the exit value of the last command in the **until** *list* is zero. If no commands in the **do** *list* are executed, the **until** command returns a zero exit value.
- (*list*) Executes the commands in *list* in a subshell.
- {*list*; } Executes the commands in *list* in the current shell process (does not spawn a subshell).
- name* () { *list*; } Defines a function that is referenced by *name*. The body of the function is the *list* of commands between the braces.

The following words are recognized only as the first word of a command and when not quoted:

**if then else elif fi case esac for while until do done {}**

## Built-in Commands

- :** Does nothing. This null command returns a zero exit value.
- . file** Reads and executes commands from *file* and returns. Does not spawn a subshell. The search path specified by **PATH** is used to find the directory containing *file*.
- break** [*n*] Exits from the enclosing **for**, **while**, or **until** loop, if any. If *n* is specified, then breaks *n* levels.
- continue** [*n*] Resumes the next iteration of the enclosing **for**, **while**, or **until** loop. If *n* is specified, resumes at the *n*th enclosing loop.
- cd** [*dir*] Changes the current directory to *dir*. The value of the shell variable **HOME** is the default *dir*. The shell variable **CDPATH** defines the search path for the directory containing *dir*. Alternative directory names appear in a colon-separated list. A null path name specifies the current directory (which is the default path). This null path name can appear immediately after the equal sign in the assignment or between the colon delimiters anywhere else in the path list. If *dir* begins with a slash, the shell does not use the search path. Otherwise, the shell searches each directory in the path. **cd** cannot be executed by the restricted shell.
- echo** [*arg* . . . ] Writes arguments to standard output. See “**echo**” on page 278 for a discussion of its usage and parameters.
- eval** [*arg* . . . ] Reads arguments as input to the shell and executes the resulting command(s).

- 
- exec** [*arg* . . . ] Executes the command specified by argument in place of this shell without creating a new process. Input and output arguments can appear and, if no other arguments appear, cause the shell input or output to be modified (not a good idea with your login shell).
- exit** [*n*] Causes a shell to exit with the exit value specified by *n*. If you omit *n*, the exit value is that of the last command executed (an end of file will also cause a shell to exit).
- export** [*name* . . . ] Marks the specified *names* for automatic export to the environments of subsequently executed commands. If you specify no *names*, **export** displays a list of all names that are exported in this shell. You **cannot** export function names.
- hash** [-r] [*name* . . . ] For each *name*, finds and remembers the location in the search path of the command specified by *name*. The -r flag causes the shell to forget all locations. If you do not specify the flag or any *names*, the shell displays information about the remembered commands. In this information, **hits** is the number of times a command has been run by the shell process. **Cost** is a measure of the work required to locate a command in the search path. There are certain situations that require that the stored location of a command be recalculated (for example, the location of a relative path name when the current directory changes). Commands for which that might be done are indicated by an asterisk next to the **hits** information. **Cost** is incremented when the recalculation is done.
- newgrp** [*arg* . . . ] Executes the **newgrp** command in the current shell process. See “**newgrp**” on page 510 for a discussion of command options.
- pwd** Displays the current directory. See “**pwd**” on page 589 for a discussion of command options.
- read** [*name* . . . ] Reads one line from standard input. Assigns the first word in the line to the first *name*, the second word to the second *name*, and so on, with leftover words assigned to the last *name*. This command returns a 0 unless it encounters an end of file.
- readonly** [*name* . . . ] Marks the specified *names* **readonly**. The values of these *names* cannot be reset. If you do not specify any *names*, **readonly** displays a list of all **readonly** names.
- return** [*n*] Cause a function to exit with a return value of *n*. If you do not specify *n*, the function returns the status of the last command executed in that function. This command is valid only when executed within a shell function.

**set** [*flag* . . . [*arg*] . . . ]

- a Marks for export all variables that are modified or changed.
- e Exits immediately if a command exits with a nonzero exit value.
- f Disables file-name substitution.
- h Locates and remembers the commands called within functions as the functions are defined (normally these commands are located when the function is executed—see the **hash** command on page 655).
- k Places all keyword parameters in the environment for a command, not just those that precede the command name.
- n Reads commands but do not execute them.
- t Exits after reading and executing one command.
- u Treats an unset variable as an error when performing variable substitution.
- v Displays shell input lines as they are read.
- x Displays commands and their arguments as they are executed.
- Does not change any of the flags. This is useful in setting \$1 to a string beginning with a - (minus).

Using a + (plus) rather than a - (minus) unsets flags. You can also specify these flags on the shell command line. The special variable \$- contains the current set of flags.

Any arguments to **set** are positional parameters and are assigned, in order, to \$1, \$2, and so on. If you specify no flags or parameters, **set** displays all names.

**shift** [*n*]

Shifts command line arguments to the left; that is, reassign the value of the positional parameters by discarding the current of value of \$1 and assigning the value of \$2 to \$1, of \$3 to \$2, and so on. If there are more than 9 command line arguments, the tenth is assigned to \$9 and any that remain are still unassigned (until after another **shift**). If there are 9 or fewer arguments, a **shift** unsets the highest-numbered positional parameter.

\$0 is never shifted. The command **shift** *n* is a shorthand notation for *n* consecutive shifts. The default value of *n* is 1.

**test** *expr* | [*expr*] Evaluates conditional expressions. See “**test**” on page 750 for a discussion of command options.

- 
- times** Displays the accumulated user and system times for processes run from the shell.
- trap** [*arg*] [*n*] . . . Runs the command specified by *arg* when the shell receives signal(s) *n*. (Note that the shell scans *parm* once when the trap is set and once when the trap is taken). **trap** commands are executed in order of signal number. Any attempt to set a trap on a signal that was ignored on entry to the current shell is ineffective. If you specify no *arg*, then all trap(s) *n* are reset to their current values. If *arg* is the null string, then this signal is ignored by the shell and by the commands it invokes. If *n* is 0, then *arg* is executed on exit from the shell. If you specify no *arg* and no *n*, **trap** displays a list of commands associated with each signal number.
- type** [*name* . . . ] For each *name*, indicates how the shell would interpret it as a command name.
- ulimit** [-**b** [*m*]] [-**f**] [*n*] [-**s** [*m*]]  
Sets or queries size limits.
- The **-b** flag sets the break value to *m*. This limits the size of data segment to *m* pages. If you specify **-b** with no *m*, **ulimit** displays the current break value.
- The **-f** flag imposes a size limit of *n* blocks on files written by the child processes (files of any size can be read). Without *n*, **ulimit** displays the current limit (this is the default action of **ulimit**).
- Note:** Since the shell rounds *n* **down** to the nearest cluster size, it is best to make it a multiple of 4 (for example, specifying values of 1, 2, or 3 for *n* all result in a size limit of 0).
- Any user can decreased this limit, but only a user operating with superuser authority can increase the limit.
- The **-s** flag imposes a size limit of *m* pages on the stack. If you specify **-s** with no *m*, **ulimit** displays the current stack size limit.
- umask** [*nnn*] Sets the user file-creation mask to *nnn* (see the **umask** system call). If you omit *nnn*, **umask** displays the current value of the mask.
- unset** [*name* . . . ] For each *name*, removes the corresponding variable or function. The variables **PATH**, **PS1**, **PS2**, **MAILCHECK** and **IFS** cannot be unset.
- wait** [*n*] Waits for the child process whose process number is *n* to end and reports its termination status. If you do not specify *n*, then the shell waits for all currently active child processes and the return value is 0 (see “**wait**” on page 844).

## Running the Shell

The **sh** command can be run either as a *login shell* or as a Only the **login** command can call **sh** as a login shell. It does this by using a special form of the **sh** command name: **-sh**. When called with an initial **-** (minus), the shell first reads and runs commands found in the system **profile** file and your **\$HOME/.profile**, if one exists. It then accepts commands as described in the following discussion of flags.

Once logged in and working under a login shell, you can call **sh** with the command name **sh**. This command runs a subshell, a second shell running as a child of the login shell.

## Restricted Shell

The restricted shell, **rsh**, is used to set up login names and environments whose capabilities are more controlled than those of the standard shell. The actions of **rsh** are identical to those of **sh**, except that the following are not allowed:

- Changing directory (see “**cd**” on page 121)
- Setting the value of **\$PATH**
- Specifying path or command names containing **/**
- Redirecting output (**>** and **>>**).

The restrictions above are enforced after **.profile** is interpreted, meaning that the restrictions can override settings originally set in **.profile**.

When a command to be run is found to be a shell procedure, **rsh** starts **sh** to run it. Thus, it is possible to provide to the end-user shell procedures that have access to the full power of the standard shell, while imposing a limited menu of commands; this scheme assumes that the end-user does not have write and run permissions in the same directory.

The net effect of these rules is that the writer of the **.profile** has complete control over user actions, by performing setup actions and leaving the user in an appropriate directory (probably not the login directory).

When called with the name **-rsh**, **rsh** reads the user’s **.profile** (from **\$HOME/.profile**). It acts as the standard **sh** while doing this, except that an interrupt causes an immediate exit instead of a return to command level.

## Flags

The following flags are interpreted by the shell only when you call it. Note that unless you specify either the **-c** or **-s** flag, the shell assumes that the next parameter is a command file (shell procedure). It passes anything else on the command line to that command file (see “Positional Parameters” on page 641).

- c** *cmdstring*  
Runs commands read from *cmdstring*. The shell does not read additional commands from standard input when you specify this flag.
- i**  
Makes the shell interactive, even if input and output are not from a work station. In this case the shell ignores the **TERMINATE** signal (so that **kill 0** does not stop an interactive shell) and traps an **INTERRUPT** (so that you can interrupt **wait**). In all cases, the shell ignores the **QUIT** signal. (See the **signal** system call in *AIX Operating System Technical Reference*, “**kill**” on page 422, and “**wait**” on page 844 for more information about signals.)
- r**  
Creates a restricted shell (the same as running **rsh**).
- s**  
Reads commands from standard input. Any remaining parameters specified are passed as positional parameters to the new shell. Shell output is written to standard error, except for the output of built-in commands (see “Built-in Commands” on page 654).

The remaining flags and parameters are described in the built-in **set** command on page 656.

## Files

/etc/environment  
/etc/profile  
\$HOME/.profile  
/tmp/sh\*  
/dev/null

## Related Information

The following commands: “**cd**” on page 121, “**echo**” on page 278, “**env**” on page 298, “**login**” on page 453, “**newgrp**” on page 510, “**pwd**” on page 589, “**test**” on page 750, “**umask**” on page 784.1, and “**wait**” on page 844.

The **dup**, **exec**, **fork**, **fullstat**, **pipe**, **signal**, **ulimit**, and **umask** system calls and the **a.out**, **.profile**, and **environ** files in *AIX Operating System Technical Reference*.

“Using the Shell with Processes” and “Advanced Shell Features” in *Using the AIX Operating System*.

“Overview of International Character Support” in *Managing the AIX Operating System*.



# shlib

---

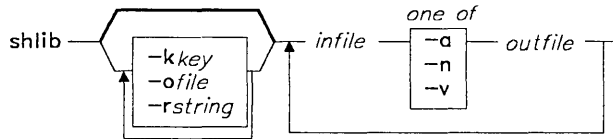
## shlib

---

### Purpose

Creates a shared library.

### Syntax



OL805448

### Description

The **shlib** command creates a shared library from a set of unshared object and/or archive files. The shared library it creates has two parts:

- A single shared library text image that contains the code, and only the code, from all of the input files.
- Modified archive or object files that refer to the text image, each of which corresponds to one of the **shlib** input files.

By default, **shlib** uses the name of the first input file to generate the shared library key. It does this by removing any directory path from the file name and, if the file name does not contain a suffix, adding the suffix **.yyddd**, where **yy** is the last two digits of the current year and **ddd** is the number of the day. **shlib** puts this key in each of the modified output files. By default, it also uses this key as the name of the shared library text image, which contains the shared library key in a form that can be found by the **what** command.

The **shlib** command transforms each input file specified on the command line and copies it to (or verifies it against) an output file. Each output object module differs from the corresponding input object module in that the text portion has been removed and added to the end of the shared library text image. **shlib** also appends the shared library key to each output module and marks its **a.out** header so the **ld** command recognizes that the file refers to a shared library and relocates references appropriately.

Once you create an archive for a shared library, you can use the **ar** command to replace individual object files. The new object files will not refer to the shared library, thus allowing you to patch shared libraries.

The **shlib** command can process all **cc** and **f77** programs. Other programs must have **KCALL** relocation entries as the only external references within the text portion (see the **a.out** file in *AIX Operating System Technical Reference*). **KCALL** relocation entries are replaced by **balax** instructions to location **0xC00**, which contain a code fragment to continue calls across segments. That code requires that register 0 point to the constant pool of the called routine, the first entry of which is the entry point of the invoked routine.

## Flags

- a** *outfile*      Adds new object modules in archives to *outfile*. This lets you add functions to a shared library, replacing the shared library text image without relinking programs that refer to it. The entire shared library image must be rebuilt.
- k***key*            Uses the shared library *key* in the output object modules to refer to the shared library text image. If *key* does not contain a . (period), a suffix in the form **.yyddd** is added.
- n** *outfile*      Makes a new output file for each input file.
- o***file*            Assigns the name *file* to the shared library text image. **shlib** always makes a new shared library text image, replacing the old one.
- r***string*        Adds *string* to the end of the shared library key in the **what** string. This only applies to the text image file.
- v** *outfile*      Verifies components of a changed shared library, thus ensuring that the resulting object files are the same as the files previously created by **shlib**. Changes can be made to C source code as long as references to external names are not added, deleted, or rearranged and no floating point, string, or static initial values are modified.

## Examples

1. To create a new shared text image:

```
shlib -kclibs -r"C shared library text" \
    /lib/libc.a -n /lib/libcs.a /lib/librts.a -n /lib/librtss.a
```

This creates shared libraries **libcs.a** and **librtss.a** and the text image file **clibs.86133** (86133 indicates that this file was produced on 13 May 1986).

2. To modify an existing shared text image:

```
shlib -kclibs -r"C shared library text" \
    /lib/libc.a -a /lib/libcs.a /lib/libm.a -n /lib/libms.a
```

## shlib

---

This updates the shared library `libcs.a`, creates the shared library `libms.a`, and updates the the shared text image `clibs.86133`. (`libc.a` may contain new members that are added to the shared library and the the text image.)

3. To create a shared text image with a different name than its key:

```
shlib -kclibs -octext_image -r"C shared library text" \  
    /lib/libc.a -n /lib/libcs.a /lib/librts.a -n /lib/librtss.a
```

This produces a text image file named `ctext_image`. Note that you must use the `-k` flag when you compile programs that use this text image, since the key and the file name are not the same:

```
cc myproj.c -kclibs:ctext_image -lrtss -lcs
```

## Related Information

The following command: “**ld**” on page 427.

The **profil** system call, **monitor** subroutine and **a.out** file in *AIX Operating System Technical Reference*.

The discussion of shared libraries in *AIX Operating System Programming Tools and Interfaces*.

---

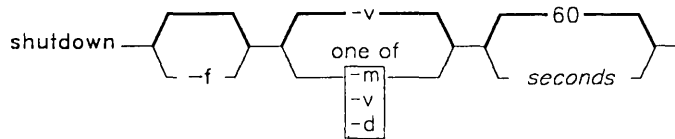
# shutdown

---

## Purpose

Ends system operation.

## Syntax



OL805215

## Description

The **shutdown** command brings the AIX Operating System from distributed mode to multiuser mode, from multiuser mode to maintenance mode, or halts the operating system completely by shutting down the virtual machine. You can run **shutdown** only if you are a member of the system group or if you have superuser authority.

During the default shutdown, all users are notified through the **wall** command of the impending system shutdown. The **hold** command prevents any new logins. After the specified number of *seconds* (60 by default), the system stops the accounting and error-logging processes. **shutdown** then runs the **killall** command to end any remaining processes and runs the **sync** command to flush all memory resident disk blocks. Finally, it unmounts the file systems and sends the appropriate signal to **init**. These signals are:

|                |                       |
|----------------|-----------------------|
| <b>SIGINT</b>  | Maintenance mode      |
| <b>SIGTERM</b> | Virtual machine halt. |

**Note:** Users who have files open on the node that is running **shutdown**, but who are not logged on that node are not notified about the shutdown.

If you request a complete halt to the operating system, **shutdown** kills all processes, unmounts all file systems, and sends **init** the **SIGTERM** signal.

**Warning:** If you are bringing the system down to maintenance mode, you must run **shutdown** from the root directory to ensure that it can cleanly unmount the file systems.

## shutdown

---

### Flags

- d Brings the system down from a distributed mode to a multiuser mode.
- f Does a fast shutdown, bypassing the messages to other users and bringing the system down as quickly as possible. If you do not specify this flag, **shutdown** sends a message to each logged-in user and waits a certain amount of time before bringing the system down, to allow each user to log off cleanly.
- m Brings the system down to maintenance mode. From maintenance mode, you can return to multi-user mode.
- v Shuts down the virtual machine (halts the operating system completely).

### Examples

1. To tell the operating system you are about to turn off the machine:

```
shutdown
```

This shuts down the system, waiting 60 seconds before stopping the user processes and the **init** process.

2. To give users and the system more time to finish what they are doing:

```
shutdown -m 120 *
```

This brings the system down from multiuser mode to maintenance mode after waiting 120 seconds.

### Files

```
/etc/shutdown.sh  
/etc/fshutdown.sh
```

### Related Information

The following commands: “**acct/\***” on page 31, “**errstop**” on page 309, “**init**” on page 396, “**kill**” on page 422, and “**killall**” on page 425,

The **dsstate** and **signal** system calls in *AIX Operating System Technical Reference*.

---

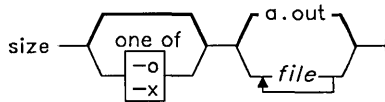
# size

---

## Purpose

Displays the section sizes of common object files.

## Syntax



OL805216

## Description

The **size** command writes to the standard output the number of bytes required by the text, initialized data, and uninitialized data, along with their sum for each *file*. The default *file* is **a.out**.

## Flags

The output is in decimal notation unless you change the output with the following flags:

- o** Writes in octal notation.
- x** Writes in hexadecimal notation.

## Examples

1. To display the size of **a.out** in decimal:

```
size
```

This displays the size in bytes of the executable file **a.out**. The size of each section of the object file is given, followed by the total. The three sections are program text, data, and bss (uninitialized data). The values are in decimal.

2. To display the size of an object file in octal:

```
size -o driver.o
```

This displays the size of the object file **driver.o** in octal.

## size

---

3. To display the size of several object files in hex:

```
size -x *.o
```

This displays in hexadecimal the size of each file in the current directory ending with .o.

## Related Information

The following commands: “**ar**” on page 58, “**as**” on page 64, “**cc**” on page 112, “**dump**” on page 275, “**ld**” on page 427, “**nm**” on page 521, and “**strip**” on page 716.

The discussion of the **a.out** and **ar** files in *AIX Operating System Technical Reference*.

# skulker

---

## Purpose

Cleans up file systems by removing unwanted files.

## Syntax

skulker **—**

OL805217

## Description

**Warning:** Because this command file runs with superuser authority and its whole purpose is to remove files, it has the potential for unexpected results. Before installing a new **skulker**, test any additions to its file removal criteria by running it manually using the **xargs -p** command. After you have verified that the new **skulker** removes only the files you want removed, you can install it.

The **skulker** command is a shell command file for periodically purging obsolete or unneeded files from file systems. Candidate files include those in **/tmp**, **.bak** files older than a specified age, and files named **a.out**, **core**, or **ed.hup**.

The **skulker** command is normally invoked daily, often as part of an accounting procedure run by **cron** during off-peak periods. Individual sites should modify **skulker** to suit local needs following the patterns shown in the distributed version. Local users should be made aware of the criteria for automatic file removal.

The **find** and **xargs** commands form a powerful combination for use in **skulker**. Most file selection criteria can be expressed conveniently with **find** expressions. The resulting file list, generated with the **-print** flag of the **find** command, can then be segmented and inserted into **rm** commands using **xargs** to reduce the overhead that would result if each file were deleted with a separate command.

## Related Information

The following commands: “**cron**” on page 172, “**find**” on page 326, “**rm**” on page 601, and “**xargs**” on page 857.



# sleep

---

## sleep

---

### Purpose

Suspends execution for an interval.

### Syntax

```
sleep — seconds —|
```

OL805218

### Description

The **sleep** command suspends execution of a process for the interval specified by *seconds*. *seconds* can range from 1 to 65,536 seconds.

### Examples

1. To run a command after a certain amount of time has passed:

```
echo "SYSTEM SHUTDOWN IN 10 MINUTES!" | wall  
(sleep 300; echo "SYSTEM SHUTDOWN IN 5 MINUTES!" | wall) &  
(sleep 540; echo "SYSTEM SHUTDOWN IN 1 MINUTE!" | wall) &  
(sleep 600; shutdown) &
```

This command sequence warns all users 10 minutes, 5 minutes, and 1 minute before the system is shut down.

2. To run a command at regular intervals:

```
while true  
do  
    date  
    sleep 60  
done
```

This shell procedure displays the date and time once a minute. To stop it, press **INTERRUPT (Alt-Pause)**.

## **Related Information**

The following commands: “**shutdown**” on page 663 and “**wall**” on page 845.

The **alarm** system call and the **sleep**, **pause**, and **signal** subroutines in *AIX Operating System Technical Reference*.

## sno

---

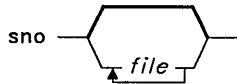
## sno

---

### Purpose

Provides a SNOBOL interpreter.

### Syntax



OL805219

### Description

The **sno** command provides a SNOBOL compiler and interpreter, with some differences from standard SNOBOL. It reads the named *files* and the standard input. It compiles all input through a statement containing the label **end**. The rest is available to **syspit**. The **sno** command differs from SNOBOL in the following ways:

- There are no unanchored searches. To get the same effect:  
a \*\* b                    Unanchored search for *b*  
a \*x\* b = x c            Unanchored assignment.
- There is no back referencing.  
x = "abc"  
a \*x\* x                    Unanchored search for **abc**.
- Function declaration is done at compile time by the use of the (nonunique) label **define**. Execution of a function call begins at the statement following the **define**. Functions cannot be defined at run time, and the use of the name **define** is preempted. There is no provision for automatic variables other than parameters. Examples:  
define f()  
define f(a, b, c)
- All labels except **define** (even **end**), must have a nonempty statement.
- Labels, functions, and variables must all have distinct names. In particular, the nonempty statement on **end** cannot merely name a label.
- If **start** is a label in the program, program execution begins there. If not, execution begins with the first executable statement. **define** is not an executable statement.
- There are no built-in functions.

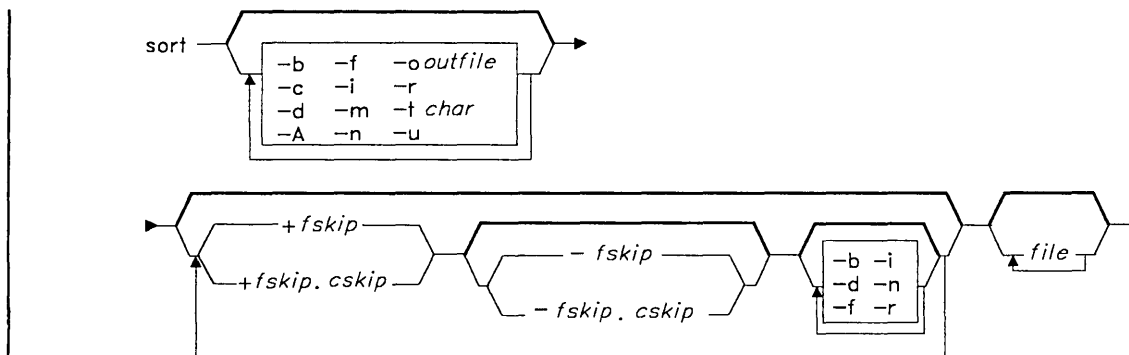
- Parentheses for arithmetic are not needed. Normal precedence applies. Because of this, the arithmetic operators \ (backslash) and \* (asterisk) must be set off by spaces.
- The right side of assignments must be nonempty.
- Either ' (single quotation mark) or " (double quotation mark) can be used for literal quotation marks.
- The pseudo-variable **sysppt** is not available.

## Related Information

The following command: “**awk**” on page 70.

**sort****sort****Purpose**

Sorts or merges files.

**Syntax**

OL805380

**Description**

The **sort** command sorts lines in its input *files* and writes the result to standard output. It treats all of its input *files* as one file when it performs the sort. A - (minus) in place of a file name specifies standard input. If you do not specify any file names, it sorts standard input.

The default sort key (the part of the line used for sorting) is an entire line. Default ordering is lexicographic by characters in the collating sequence. The file `/usr/pub/ascii` shows the default collating sequence. To change the default collating sequence, see “**ctab**” on page 204.

The two numbers, *fskip* and *cskip*, specify the sort key. Both numbers have two parts, as follows:

`+fskip.cskip`  
`-fskip.cskip`

The *fskip* specifies the number of fields to skip from the beginning of the input line, and *cskip* specifies the number of additional characters to skip to the right beyond that point. For both the starting point (`+fskip.cskip`) and the ending point (`-fskip.cskip`) of a sort key,

*fskip* is measured from the beginning of the input line, and *cskip* is measured from the last field skipped. If you omit *.cskip*, *.0* is assumed. If you omit *fskip*, *0* is assumed. If you omit the ending field specifier (*-fskip.cskip*), the end of the line is the end of the sort key.

You can supply more than one sort key by repeating *+fskip.cskip* and *-fskip.cskip*. In cases where you specify more than one sort key, keys specified further to the right on the command line are compared only after all earlier keys are sorted. For example, if the first key is to be sorted in numerical order and the second in dictionary order, all strings that start with the number one are sorted alphabetically before the strings that start with the number two. Lines that are identical in all keys are sorted with all characters significant.

You can also specify different flags for different sort keys in multiple sort keys. See the examples for illustration.

A field is one or more characters bounded by the beginning of a line and the current field separator, or one or more characters bounded by a the field separator on either side. The space character is the default field separator.

**Note:** Very long lines are truncated silently.

## Flags

- A** Sorts on a byte-by-byte basis. This sort is functionally compatible with the Version 1.1 **sort** command, prior to the addition of international character support.
- b** Ignores leading blanks, spaces, and tabs in sort key comparisons.
- c** Checks that the input is sorted according to the ordering rules specified in the flags. Displays nothing unless the is not sorted.
- d** Sorts in dictionary order. Only letters, digits and blanks are considered in comparisons.
- f** Merges uppercase and lowercase letters. Case is not considered in the sorting, so that initial-capital words and all-capital words are not grouped together at the beginning of the output.
- i** Sorts only by characters in the ASCII range octal 040-0176 (all printable characters and the space character) in nonnumeric comparisons.
- m** Merges only; the input is already sorted.
- n** Sorts any initial numeric strings (consisting of optional blanks, optional minus signs, and zero or more digits with optional decimal point) by arithmetic value. The **-n** flag automatically gives you the **-b** flag.
- o *outfile*** Directs output to *outfile* instead of standard output. *outfile* can be the same as one of the input *files*.
- r** Reverses the order of the specified sort.

## sort

---

- tchar** Sets field separator character to *char*. To specify the tab character as the field separator, you must enclose it in single quotation marks (' ').
- u** Suppresses all but one in each set of equal lines. Ignored characters (such as leading tabs and spaces) and characters outside of sort keys are not considered in this type of comparison.

## Examples

1. To perform a simple sort:

```
sort fruits
```

This displays the contents of `fruits` sorted in ascending lexicographic order. This means that the characters in each column are compared one by one, including spaces, digits, and special characters. For instance, if `fruits` contains the text:

```
banana  
apple  
orange  
Persimmon  
apple  
%%banana  
pear  
ORANGE
```

then **sort** displays:

```
%%banana  
ORANGE  
Persimmon  
apple  
apple  
banana  
orange  
pear
```

This order follows from the fact that in the ASCII collating sequence, % (percent sign) precedes the uppercase letters, which precede the lowercase letters. If the system uses a character set other than ASCII, your results may be different.

2. To sort in dictionary order:

```
sort -d fruits
```

This sorts and displays the contents of `fruits`, comparing only letters, digits, and blanks. If `fruits` is the same as in Example 1, then **sort** displays:

```
ORANGE
Persimmon
apple
apple
%%banana
banana
orange
pear
```

The `-d` flag tells **sort** to ignore the `%` character because it is not a letter, digit, or blank. This puts `%%banana` next to `banana`.

3. To group lines that contain uppercase and special characters with similar lowercase lines:

```
sort -d -f fruits
```

This ignores special characters (`-d`) and differences in case (`-f`). Given the `fruits` of Example 1, this displays:

```
apple
apple
%%banana
banana
ORANGE
orange
pear
Persimmon
```

4. To sort as in Example 3 and remove duplicate lines:

```
sort -d -f -u fruits
```

The `-u` flag tells **sort** to remove duplicate lines, making each line of the file unique. This displays:

```
apple
%%banana
orange
pear
Persimmon
```



## sort

---

Note that not only was the duplicate apple removed, but banana and ORANGE as well. These were removed because the `-d` told **sort** to treat `%%banana` as if it were banana, and the `-f` told it to treat ORANGE as orange. Thus, **sort** considered `%%banana` to be a duplicate of banana and ORANGE a duplicate of orange.

**Note:** There is no way to predict which duplicate lines `sort -u` will keep and which it will remove.

5. To sort as in Example 3 and remove duplicates, unless capitalized or punctuated differently:

```
sort -u +0 -d -f +0 fruits
```

The `+0 -d -f` does the same type of sort done with `-d -f` in Example 3. Then the `+0` performs another comparison to distinguish lines that are not actually identical. This prevents `-u` from removing them.

Given the `fruits` file shown in Example 1, the added `+0` distinguishes `%%banana` from banana and ORANGE from orange. However, the two instances of apple are identical, so one of them is deleted.

```
apple
%%banana
banana
ORANGE
orange
pear
Persimmon
```

6. To specify the character that separates fields:

```
sort -t: +1 vegetables
```

This sorts `vegetables`, comparing the text that follows the first colon on each line. The `+1` tells **sort** to ignore the first field and to compare from the start of the second field to the end of the line. The `-t:` tells **sort** that colons separate fields. If `vegetables` contains:

```
yams:104
turnips:8
potatoes:15
carrots:104
green beans:32
radishes:5
lettuce:15
```

then **sort** displays:

```
carrots:104
yams:104
lettuce:15
potatoes:15
green beans:32
radishes:5
turnips:8
```

Note that the numbers are not in numeric order. This happened because a lexicographic sort compares each character from left to right. In other words, “3” comes before “5” and “2” comes before “ ”, so “32” comes before “5 ”.

7. To sort numbers:

```
sort -t: +1 -n vegetables
```

This sorts `vegetables` numerically on the second field. If `vegetables` is the same as in Example 6, then **sort** displays:

```
radishes:5
turnips:8
lettuce:15
potatoes:15
green beans:32
carrots:104
yams:104
```

8. To sort on more than one field:

```
sort -t: +1 -2 -n +0 -1 -r vegetables
```

This performs a numeric sort on the second field (+1 -2 -n). Within that ordering, it sorts the first field in reverse alphabetic order (+0 -1 -r). The output looks like this:

```
radishes:5
turnips:8
potatoes:15
lettuce:15
green beans:32
yams:104
carrots:104
```

Now the lines are sorted in numeric order. When two lines have the same number, they appear in reverse alphabetic order.

## sort

---

9. To replace the original file with the sorted text:

```
sort -o vegetables vegetables
```

This stores the sorted output into the file `vegetables` (`-o vegetables`).

## Files

`/usr/tmp/stm???`

## Related Information

The following commands: “**comm**” on page 144, “**join**” on page 417, and “**uniq**” on page 792.

The “Overview of International Character Support” in *Managing the AIX Operating System*.

---

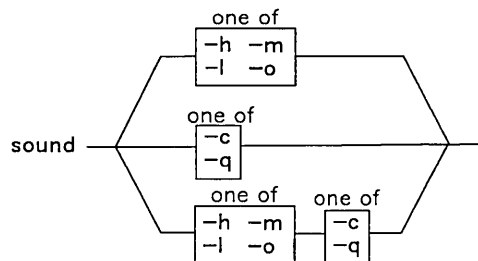
# sound

---

## Purpose

Controls the volume and click of the keyboard speaker.

## Syntax



OL805415

## Description

The **sound** command controls the volume of the sound output (the console bell and the keyboard click) and, additionally, whether or not the keyboard click is produced. You can modify these two sound characteristics independently of each other.

The system startup process sets the sound volume to medium.

**Note:** You can run **sound** only from the console (`/dev/console`).

## Flags

You must select at least one flag from the following two groups of flags or, optionally, one flag from each of the two groups. The first group of flags controls the volume of all sound output:

- h** Sets the volume to high.
- l** Sets the volume to low.
- m** Sets the volume to medium.
- o** Turns the volume off.

## sound

---

The second group of flags controls whether or not click sounds are produced:

- c Turns clicking on.
- q Turns clicking off (quiet).

### Example

To set the volume to low and turn the click function on:

```
sound -lc
```

In addition to turning on the keyboard click (-c), this command sets the volume of both the bell and the click to low (-l).

---

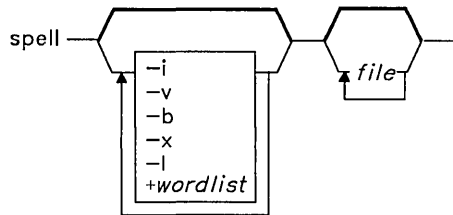
# spell

---

## Purpose

Finds spelling errors.

## Syntax



`/usr/lib/spell/hashmake`  $\rightarrow$

`/usr/lib/spell/spellin`  $\rightarrow$  *num*  $\rightarrow$

`/usr/lib/spell/hashcheck`  $\rightarrow$  *spellinglist*  $\rightarrow$

OL805304

## Description

The **spell** command reads words in *file* and compares them to those in a spelling list. Words that cannot be matched in the spelling list or derived from words in the spelling list (by applying certain inflections, prefixes, and/or suffixes) are written to standard output. If you do not specify a file to read, **spell** reads standard input.

The **spell** command ignores the same **troff**, **tbl**, and **eqn** constructs as the **deroff** command.

The coverage of the spelling list is uneven. You should create your own dictionary of special words used in your files.

Certain auxiliary files can be specified by file name parameters; see “Files” on page 683. Copies of all output are accumulated in the history file.

Three routines help maintain and check the hash lists used by **spell**.

# spell

---

|                                                           |                                                                                                                                                   |
|-----------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>/usr/lib/spell/hashmake</code>                      | Reads a list of words from standard input and writes the corresponding nine-digit hash code to standard output.                                   |
| <code>/usr/lib/spell/spellin <i>num</i></code>            | Reads <i>num</i> hash codes from standard input and writes a compressed spelling list to standard output.                                         |
| <code>/usr/lib/spell/hashcheck <i>spellinglist</i></code> | Reads a compressed <i>spellinglist</i> and recreates the nine-digit hash codes for all the words in it; it writes these codes to standard output. |

## Flags

|                                |                                                                                                                                                                                                                                                                                                     |
|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>-b</code>                | Checks British spelling.                                                                                                                                                                                                                                                                            |
| <code>-i</code>                | Suppresses processing of included files.                                                                                                                                                                                                                                                            |
| <code>-l</code>                | Follows the chain of all included files ( <code>.so</code> and <code>.nx</code> formatting commands). Without this flag, <code>spell</code> follows chains of all included files except for those beginning with <code>/usr/lib</code> .                                                            |
| <code>-v</code>                | Displays all words not literally in the spelling list and indicates plausible derivations from the words.                                                                                                                                                                                           |
| <code>-x</code>                | Displays every plausible word stem with an = (equal sign).                                                                                                                                                                                                                                          |
| <code>+ <i>wordlist</i></code> | Checks <i>wordlist</i> for additional word spellings. <i>wordlist</i> is the name of a file you provide that contains a sorted list of words, one per line. With this flag, you can specify a set of correctly spelled words (in addition to <code>spell</code> 's own spelling list) for each job. |

## Examples

1. To check your spelling:

```
spell chap1 >mistakes
```

This creates a file named `mistakes` containing all the words found in `chap1` that are not in the system spelling dictionary. Some of these may be correctly spelled words that `spell` does not know about. It is a good idea to save the output of `spell` in a file because the word list may be long.

2. To check British spelling:

```
spell -b chap1 >mistakes
```

This checks `chap1` against the British dictionary and writes the questionable words in `mistakes`.

3. To see how **spell** derives words:

```
spell -v chap1 >deriv
```

This lists the words that are not found literally in the dictionary, but are derived forms of dictionary words. The prefixes and suffixes used to form the derivative are indicated for each word. Words that do not appear in the dictionary at all are also listed.

4. To check your spelling against an additional word list:

```
spell +newwords chap1
```

This checks the spelling of words in `chap1` against the system dictionary and against `newwords`. The file `newwords` lists words in alphabetical order, one per line. You can create this file with a text editor, such as **ed**, and alphabetize it with the **sort** command.

## Files

|                                               |                                                        |
|-----------------------------------------------|--------------------------------------------------------|
| <code>D_SPELL=/usr/lib/spell/hlist[ab]</code> | Hashed spelling lists, American and British.           |
| <code>S_SPELL=/usr/lib/spell/hstop</code>     | Hashed stop list.                                      |
| <code>H_SPELL=/usr/lib/spell/spellhist</code> | History file.                                          |
| <code>/usr/lib/compress</code>                | Executable shell program to compress the history file. |
| <code>/usr/lib/spell/spellprog</code>         | Program.                                               |

## Related Information

The following commands: “**deroff**” on page 239, “**eqn**, **neqn**, **checkeq**” on page 300, “**sed**” on page 629, “**sort**” on page 672, “**tbl**” on page 739, “**tee**” on page 746, and “**troff**” on page 526.



# spline

---

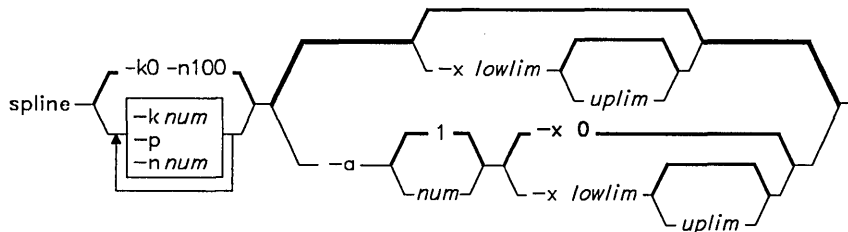
## spline

---

### Purpose

Interpolates smooth curve.

### Syntax



OL805261

### Description

The **spline** command reads from the standard input pairs of numbers that represent the coordinates of a point on an x,y axis. From this input, **spline** calculates the coordinates of points to form a smooth curve through the points in the input set. It then writes these points to standard output. The output points are approximately equally spaced and includes the points that you provided as input. The cubic spline output has two continuous derivatives, and enough points so that when plotted with the **graph** command it looks smooth.

When data is not strictly monotone in *x*, **spline** reproduces the input without interpolating extra points.

You can only use 1,000 input points.

### Flags

- anum**               Supplies abscissas automatically; spacing is given by the next parameter or is assumed to be 1 if the next parameter is not a number.
- knum**               Uses the constant *num* in the boundary value calculation:  
$$y_0 = ky_1, y = numy$$

The default for *num* is zero.

- nnum** Spaces output points so that approximately *num* intervals occur between the lower and upper *x* limits (set with the **-x** flag). The default *num* is 100.
- p** Makes output periodic, that is, matches derivatives at ends. First and last input values should normally agree.
- xlowlim[uplim]** Sets lower and upper *x* limits as *lowlim* and *uplim*. Normally, these limits are calculated from the data. Automatic abscissas start at lower limit, defaults to zero.

## Related Information

The following command: “**graph**” on page 375.

# split

---

## split

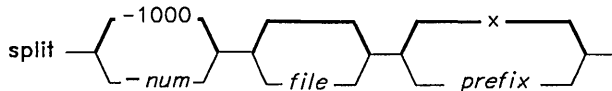
---

### Purpose

Splits a file into pieces.

### Syntax

```
split -num file prefix
```



OL805262

### Description

The **split** command reads *file* and writes it in *num*-line pieces (default 1000 lines) to a set of output files. The name of the first output file is *prefixaa*, the second is *prefixab*, and so on lexicographically, through *prefixzz* (a maximum of 676 files). *prefix* cannot be longer than 12 characters. If you do not specify an output name, **x** is assumed.

If you do not specify an input file, or if you specify - (minus) in place of *file*, then **split** reads standard input.

### Examples

1. To split a file into 1000-line segments:

```
split book
```

This splits *book* into 1000-line segments named **xaa**, **xab**, **xac**, and so forth.

2. To split a file into 50-line segments and specify the file name prefix:

```
split -50 book sect
```

This splits *book* into 50-line segments named **sectaa**, **sectab**, **sectac**, and so forth.

### Related Information

The following commands: “**bfs**” on page 90 and “**csplit**” on page 202.

---

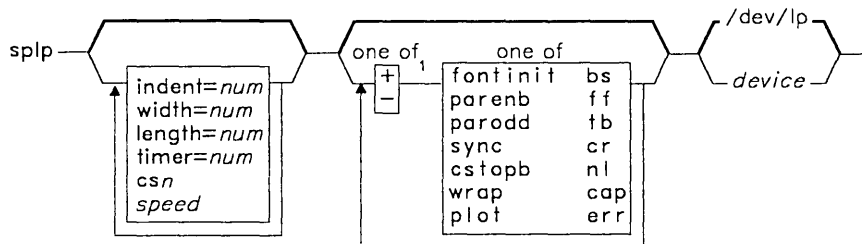
# splp

---

## Purpose

Changes or displays printer driver settings.

## Syntax



OL805263

<sup>1</sup> Do not put a blank between these items.

OL805308

## Description

The **splp** command changes or displays settings for a printer driver (*device*). The default *device* is **/dev/lp0**. If you do not specify any flags, **splp** reports the current settings for the specified *device*. Select flags to change the current settings. No other processing is done, and there is no other output.

The changes that **splp** makes remain in effect until the next time you restart the system or rerun **splp**. You can run **splp** from the **/etc/rc** command file to configure your printer each time you start up the system.

## Flags

- timer = num** Sets the time out period to *num* seconds, where *num* is an integer.
- indent = num** Indents *num* columns, where *num* is an integer.
- length = num** Prints *num* lines per page, where *num* is an integer.
- width = num** Prints *num* columns, where *num* is an integer
- + bs (-bs)** Sends (does not send) backspaces to the printer.

**splp**

---

- + **cap** (-**cap**)    Converts (does not convert) all lowercase characters to uppercase.
- + **cr** (-**cr**)        Sends carriage returns (translates carriage returns to line-feeds).
- + **cstopb** (-**cstopb**)  
                      Selects 2 (1) stop bits per character.
- cs5 cs6 cs7 cs8**  
                      Selects character size. See **termio** in *AIX Operating System Technical Reference* for additional information on character size.
- + **err** (-**err**)        Issues (does not issue) a signal (SIGIOINT) upon receiving a VRM error and attempts to resume I/O.
- + **ff** (-**ff**)         Sends form-feeds (simulates a form-feed with line-feeds or carriage returns).
- + **fontinit** (-**fontinit**)  
                      Indicates that fonts are (are not) loaded. Use this flag to control the initialization of fonts for the 3812 Pageprinter.
- + **nl** (-**nl**)         Sends line-feeds (translates line-feeds to carriage returns).
- + **parenb** (-**parenb**)  
                      Enables (disables) parity generation and detection.
- + **parodd** (-**parodd**)  
                      Selects odd (even) parity.
- + **plot**             Sends all characters to the printer unmodified. This overrides other settings.
- plot**             Translates characters according to the settings.
- + **sync** (-**sync**)  
                      Does not (does) return immediately without waiting for all data to be sent out.
- + **tb** (-**tb**)         Expands (does not expand) tabs on eight position boundaries.
- + **wrap** (-**wrap**)  
                      Wraps (truncates) characters beyond the specified **width** to the next line and (with +**wrap**), prints " . . . " before the new-line character.
- 50 75 110 134 150 300 600 1200 1800 2400 4800 9600 exta extb**  
                      Sets the *speed* to the specified number of bits per second (**exta** is 19200).

## Examples

1. To display the current printer settings:  
    splp

2. To change the printer settings:

```
splp width=80 +wrap +cap
```

This changes the settings of the `/dev/lp` printer for 80-column paper (**width=80**). It wraps each line that is more than 80 columns wide onto a second line (**+wrap**), and prints all alphabetic characters in uppercase (**+cap**).

## Related Information

The following command: “**lp**” on page 459.

The **lp** file in *AIX Operating System Technical Reference*.

### Purpose

Provides tools for analyzing numerical data.

### Description

The **stat** commands, residing in `/usr/bin/graf`, provide a package of tools for analyzing data. All numerical data are stored in vectors. A **vector** is a sequence of numbers separated by delimiters, where a number has the form:

$$[sign](digits)(.digits)[e[sign]digits]$$

Fields surrounded by brackets are optional; one or both of those surrounded by parentheses are required. Any input character that is not part of a number is assumed to be a delimiter.

Vectors are text strings that can be stored in text files and created and modified by text editors.

**Note:** Some commands limit the size of an input vector.

These commands can be divided into four classes:

- Those that produce an output vector based upon definable parameters (generators).
- Those that operate upon an input vector and output the resulting value (transformers).
- Those that perform mathematical or statistical operations on vectors (summarizers).
- Those that convert vectors into a format that can be viewed pictorially (translators).

The following parameters are used to designate the expected type of the value:

*c*      A character value.

*i*      An integer value.

*f*      A floating-point or integer value.

*file*    A file name.

*vector* A vector taken from standard input or the name of a file containing a vector. Except for the **gas**, **prime**, and **rand** commands, all of the commands discussed under **stat** read vectors from standard input (by default) or from text files as specified on the command line. A file name of - (minus) specifies standard input in a file list.

*string* A character string (quoted if it includes white space).

---

## Commands That Produce Definable Vectors (Generators)

### gas

#### *Syntax*

gas [-ci,if,ni,sf,tf]

#### *Description*

The **gas** command produces an additive sequence.

#### *Flags*

- ci Specifies the number of columns per line of output (5 by default).
- if Specifies the increment between successive elements (1 by default).
- ni Specifies the number of elements in the vector (10 by default).
- sf Specifies the starting point of the sequence (1 by default).
- tf Specifies the end of the sequence (infinity by default).

#### *Examples*

1. To generate the numbers 1 through 10:  
gas
2. To generate the sequence .01 .02 .03 .04 .05:  
gas -s.01,t.05,i.01
3. To generate the sequence 3 5 3 5:  
gas -s3,t5,i2,n4



## stat

---

### prime

#### *Syntax*

prime [-ci,hi,li,ni]

#### *Description*

The **prime** command generates consecutive prime numbers.

#### *Flags*

- ci Specifies the number of columns per line of output (5 by default).
- hi Specifies the high boundary (infinity by default).
- li Specifies the low boundary (2 by default).
- ni Specifies the number of elements in the sequence (10 by default).

#### *Example*

To generate all prime numbers between 200 and 300:

```
prime -l200,-h300
```

### rand

#### *Syntax*

rand [-ci,hf,lf,mf,ni si]

#### *Description*

The **rand** command generates a random sequence of numbers.

#### *Flags*

- ci Specifies the number of columns per line of output (5 by default).
- hf Specifies the high boundary (1 by default).
- lf Specifies the low boundary (0 by default).

- ni** Specifies the number of elements in the sequence (10 by default).
- si** Specifies a seed number (1 by default).

### ***Example***

To produce a random sequence:

```
rand
```

This generates ten random numbers between 0 and 1.

```
rand -l10,m25,c3
```

This generates ten random numbers between 10 and 35, three per line.

## **Commands That Map Input to Output (Transformers)**

### **abs**

#### ***Syntax***

```
abs [-ci] [vector . . . ]
```

#### ***Description***

The **abs** command provides the absolute value of a number.

#### ***Flag***

- ci** Specifies the number of columns per line of output (5 by default).

### ***Example***

To obtain the absolute value of each element in a vector:

```
abs -c3 myfile
```

This produces the absolute value of each number in the file `myfile` and displays these values three per line.

## af

### *Syntax*

**af** [-ci,t,v] *expression* . . .

### *Description*

The **af** command performs arithmetic operations on numbers.

### *Expressions*

Expression operands are:

**Vectors** File names with the restriction that they must begin with a letter and be composed only of letters, digits, and the `_` (underscore) and `.` (dot) characters. The first unknown file name (one not in the current directory) references standard input.

**Functions** The name of a command followed by the command arguments in parentheses. List arguments as you would on the command line.

**Constants** Floating-point and integer numbers (but not E notation).

Expression operators are, in order of decreasing precedence:

`'v` The next value from vector *v*.

`x^y` `-x` The value *x* raised to the power *y*; the negation of *x*. Both associate right to left.

`x*y` `x/y` `x%y` The value *x* multiplied by, divided by, modulo *y*, respectively. All associate left to right.

`x+y` `x-y` The value *x* plus or minus *y*. Both associate left to right.

`x,y` The value of *x* followed by the value of *y*. This associates from left to right.

You can use parentheses to alter precedence. Because many of the operator characters are special to the shell, it is good practice to quote expression arguments.

### *Flags*

`-ci` Specifies the number of columns per line of output (5 by default).

`-t` Cause the output to be titled from the vector on standard input.

`-v` Echoes function expansions.

---

### ***Examples***

1. To perform arithmetic operations:

```
af "3+4*5"
```

This yields 23.

2. To produce a matrix:

```
af "A, 'A,A+'A,B"
```

This yields a four-column matrix with columns of:

- a. odd elements from vector A
  - b. even elements from A
  - c. sum of adjacent odd and even elements from A
  - d. elements from vector B.
3. To use functions:

```
af "sin (A)^2"
```

This yields the square of the sin of the elements of vector A.

## **ceil**

### ***Syntax***

```
ceil [-ci] [vector . . . ]
```

### ***Description***

The **ceil** command rounds a number up to the next integer.

### ***Flag***

**-ci** Specifies the number of columns per line of output (5 by default).

## stat

---

### cusum

#### *Syntax*

`cusum [-ci] [vector . . . ]`

#### *Description*

The **cusum** command calculates a cumulative sum. Output is a vector with the *i*th element being the sum of the first *i* elements from the input vector.

#### *Flag*

`-ci` Specifies the number of columns per line of output (5 by default).

### exp

#### *Syntax*

`exp [-ci] [vector . . . ]`

#### *Description*

The **exp** command provides the exponential function. Output is a vector with elements *e* raised to the *x* power, where *e* is approximately 2.71828 and *x* is each element in the input vector.

#### *Flag*

`-ci` Specifies the number of columns per line of output (5 by default).

### floor

#### *Syntax*

`floor [-ci] [vector . . . ]`

#### *Description*

The **floor** command rounds a number down to the nearest integer.

**Flag**

**-ci** Specifies the number of columns per line of output (5 by default).

**gamma****Syntax**

gamma [-ci] [*vector* . . . ]

**Description**

The **gamma** command provides the gamma function.

**Flag**

**-ci** Specifies the number of columns per line of output (5 by default).

**list****Syntax**

list [-ci,*dstring* [ *file* . . . ]

**Description**

The **list** command lists vector elements.

**Flags**

**-ci** Specifies the number of columns per line of output (5 by default).

**-dstring** Specifies delimiters characters. If you do not specify **-d**, any character that is not part of a number is considered a delimiter. If you specify **-d**, the space, tab, and new-line characters, plus the characters in *string* are delimiters.

Only numbers surrounded by delimiters are listed.

### *Examples*

1. To output each element:

```
list -c3 myfile
```

This outputs each element in `myfile`, three per line.

2. To specify delimiters:

```
list -d\\, myfile
```

This outputs each element of `myfile` that is delimited by commas or white space, five per line. A comma requires two backslashes because it is a special character for `list`.

## log

### *Syntax*

```
log [-ci,-bf] [vector ...]
```

### *Description*

The `log` command provides the logarithmic function.

### *Flag*

- ci Specifies the number of columns per line of output (5 by default).
- bf Specifies the base (e by default).

### *Example*

To calculate a logarithm:

```
log -b2,c3 mydata
```

This outputs the logarithm base 2 of each element in `mydata`, three per line.

---

**mod*****Syntax***

**mod** [-ci mf] [vector . . .]

***Description***

The **mod** command returns the modulo. The output is a vector with each element being the remainder of dividing the corresponding element from the input vector by the modulus.

**-ci** Specifies the number of columns per line of output (5 by default).

**-mf** Specifies the modulus (2 by default).

***Example***

To output remainders:

```
mod -m8,c3 mydata
```

This outputs the elements of mydata modulo 8, three per line.

**pair*****Syntax***

**pair** [-ci,Ffile,xi] [vector . . .]

***Description***

The **pair** command pairs elements. Output is a vector with elements taken alternatively from a base vector and from another input vector.

***Flags***

**-ci** Specifies the number of columns per line of output (5 by default).

**-Ffile** The file containing the base vector. If you do not specify **-F**, then the base vector comes from standard input. If both the base vector and the paired vector come from standard input, the base vector precedes the paired vector.

**-xi** The number of elements per group in the base vector (1 by default).



## stat

---

### *Example*

To pair elements:

```
pair -x3,Fbasefile datafile
```

This outputs a vector with three elements from basefile, one from datafile, three from basefile, one from datafile, and so on.

## power

### *Syntax*

```
power [-ci,pf] [vector . . . ]
```

### *Description*

The **power** command raises a number to a power.

### *Flag*

- ci Specifies the number of columns per line of output (5 by default).
- pf Specifies the power (2 by default).

## root

### *Syntax*

```
root [-ci,rf] [vector . . . ]
```

### *Description*

The **root** command takes the root of a number.

### *Flags*

- ci Specifies the number of columns per line of output (5 by default).
- rf Specifies the root (2 by default).

---

**round*****Syntax***

**round** [-ci,pi,si] [vector . . . ]

***Description***

The **round** command rounds a number to the nearest integer (.5 rounds to 1).

***Flags***

- ci Specifies the number of columns per line of output (5 by default).
- pi Specifies the number of places after the decimal point (0 by default).
- si Specifies the number of significant digits.

***Example***

To round numbers to two significant digits:

```
round -s2,c3 mydata
```

**siline*****Syntax***

**siline** [-ci,if,ni,sf] [vector . . . ]

***Description***

The **siline** command generates a line, given slope and intercept.

***Flags***

- ci Specifies the number of columns per line of output (5 by default).
- if Specifies the intercept (0 by default).
- ni Specifies the number of positive integers.
- sf Specifies the slope of the line.

***Example***

To output a linear fit:

```
siline -\lref -o,FA B` A
```

This outputs a simple linear fit of vector B on vector A (The **o** flag of **lreg** outputs the slope and intercept in option form of B regressed on A.)

**sin*****Syntax***

```
sin [-ci] [vector . . . ]
```

***Description***

The **sin** command provides the sine function.

***Flags***

**-ci** Specifies the number of columns per line of output (5 by default).

**subset*****Syntax***

```
subset [-af,bf,ci,Ffile,ii,lf,nl,np,pf,si,ti] [vector . . . ]
```

***Description***

The **subset** command produces a subset of the numbers in a vector.

***Flags***

**-af** Specifies the number above which subset members are selected.

**-bf** Specifies the number below which subset members are selected.

**-ci** Specifies the number of columns per line of output (5 by default).

**-Ffile** Specifies the file containing the master vector.

**-ii** Specifies the increment between successive elements (1 by default).

- lf**     The number of elements to leave.
- nli**    Specifies that the master file contains element numbers to leave.
- npi**    Specifies that the master file contains element numbers to pick.
- pf**     The number of elements to pick.
- sf**     Specifies the starting point of the sequence (1 by default).
- tf**     Specifies the end of the sequence (32,767 by default).

### ***Examples***

1. To specify the even elements of a vector:

```
subset -i2,s2 myfile
```

2. To specify corresponding elements:

```
subset -FB,p1 A
```

For each element in B with a 1, output the corresponding element in A.

## **Commands That Calculate Statistics (Summarizers)**

### **bucket**

#### ***Syntax***

```
bucket [-ai,ci,Ffile,hf,ii,lf,ni] [vector . . . ]
```

#### ***Description***

The **bucket** command groups numbers into buckets. The input vector must be sorted. The output vector consists of odd (parenthesized) elements that are bucket limits and even elements that are bucket counts. The count is the number of elements greater than or equal to the lowest limit and less than or equal to the higher limit. Unless otherwise specified, bucket limits are generated based on the input data and the following rule:

$$\#buckets = 1 + \log_2(\#elements)$$

#### ***Flags***

- ai**     Specifies the average size of the bucket.

## stat

---

- ci** Specifies the number of columns per line of output (5 by default).
- Ffile** Specifies the file containing bucket boundaries.
- hf** Specifies the high boundary (by default, the largest element in the input vector).
- ii** Specifies the interval between successive elements.
- li** Specifies the low boundary (by default, the smallest element in the input vector).
- ni** Specifies the number of buckets.

### *Example*

To divide elements into buckets:

```
bucket -a12,1-5 myfile
```

This outputs limits and counts for the elements of `myfile`, where the lowest limit is -5 and the average bucket count is 12.

## cor

### *Syntax*

```
cor [-Ffile] [vector ...]
```

### *Description*

The `cor` command provides the ordinary correlation coefficient. Use the **F** flag to specify the base vector; otherwise it is assumed to come from standard input. Each *vector* is compared to the base vector (both must be of the same length).

### *Flag*

**-Ffile** Specifies the file containing base vector.

### *Example*

To obtain correlation coefficients:

```
cor -Ffilea olddata newdata
```

This outputs the ordinary correlation coefficients between vectors `filea` and `olddata` and vectors `filea` and `newdata`.

---

**hilo**

**hilo** [-h,l,o,ox,oy] [*vector* . . . ]

**Description**

The **hilo** command finds high and low values across all of the input vectors.

- h Finds the high value only.
- l Finds the low value only.
- o Outputs the high and low values in option form (suitable for **plot**).
- ox Outputs the high and low values in option form with **x** prefixed.
- oy Outputs the high and low values in option form with **y** prefixed.

**Example**

To find the lowest value:

```
hilo -ox,l file1 file2
```

This finds the lowest value in vectors `file1` and `file2` and outputs it with `xl` prefixed to it.

**lreg****Syntax**

**lreg** [-F*file*,i,o,s] [*vector* . . . ]

**Description**

The **lreg** command provides linear regression. Output is the slope and intercept from a least squares linear regression of each vector on a base vector. The default base vector is the ascending positive integers from zero.

**Flags**

- F*file* Specifies a file containing the first vector.
- i Outputs only the intercept.
- o Outputs the slope and intercepts in option form (suitable for **siline**).

## stat

---

**-s** Outputs only the slope.

### *Example*

To output only the intercept:

```
lreg -Fbase,i mydata
```

This outputs the intercept from the linear regression of vector `mydata` on base vector `base`.

## mean

### *Syntax*

```
mean [-ff,ni,pf ]
```

### *Description*

The **mean** command calculates the (trimmed) arithmetic mean.

### *Flags*

**-ff** Specifies the fraction of elements to trim from each end. This is calculated as follows:

$$(1/f) k$$

where  $k$  is the total number of elements.

**-ni** Specifies the number of elements to trim from each end.

**-pf** Specifies the percentage of elements to trim from each end.

### *Example*

To output the mean:

```
mean -p.25 mydata
```

This outputs the mean of the middle 50% of the elements of `mydata`; that is, `mydata` is trimmed by 25% of its elements from both ends.

---

**point*****Syntax***

`point [-ff,ni,pf,s] [vector . . . ]`

***Description***

Output from the **point** command is a linearly interpolated value from the empirical cumulative density function for the input vector. By default, **point** returns the median (50% point).

***Flags***

- ff** Returns the  $(1/f)*100$  percent point.
- ni** Returns the  $i$ th element.
- pf** Returns the  $f*100$  percent point.
- s** Specifies that the input has been sorted.

***Example***

To output the 25% point:

```
point -s,p.25 mydata
```

**prod*****Syntax***

`prod`

***Description***

The **prod** commands calculates an internal product. Output is the product of the elements in the input vectors.



## stat

---

### qsort

#### *Syntax*

**qsort** [-ci] [*vector* . . . ]

#### *Description*

The **qsort** command does a quick sort. Output is a vector of the elements from the input vector in ascending order.

#### *Flag*

**-ci** Specifies the number of columns per line of output (5 by default).

### rank

#### *Syntax*

**rank** [*vector* . . . ]

#### *Description*

The **rank** command ranks vectors. Output is the number of elements in each input vector.

### total

#### *Syntax*

**total** [*vector* . . . ]

#### *Description*

The **total** command calculates a sum total. Output is the sum total of the elements in the input vector(s).

---

**var*****Syntax***

**var** [*vector* . . . ]

***Description***

The **var** command calculates the variance.

## Commands That Produce Pictorial Output (Translators)

Input to these commands can be either a vector or a GPS object (a format for storing a picture). A picture is defined in a Cartesian plane of 64K points on each axis. The plane, or universe, is divided into 25 square regions numbered 1 to 25 from the lower left to the upper right.

**bar*****Syntax***

**bar** [-a,b,f,g,ri,wi,xf,xa,yf,ya,ylf,yhf] [*vector* . . . ]

***Description***

The **bar** command builds a bar chart. It operates on an input vector, each element of which defines the height of a bar (y-axis). By default, the x-axis is labeled with positive integers, beginning at 1. For other labels, see **label**.

***Flags***

- a Suppresses the axes.
- b Plots the bar chart with bold weight lines (medium is the default weight).
- f Does not build a frame around the plot area.
- g Suppresses the background grid.
- ri Puts the bar chart in GPS region *i*, where *i* is between 1 and 25 inclusive (13 by default).
- wi Specifies the ratio of the bar width to center-to-center spacing expressed as a percentage (50 by default, giving equal bar width and bar space).

## stat

---

- xf*
- yf*    Positions the bar chart in the GPS universe with the x-origin (y-origin) at *f*.
- xa*
- ya*    Does not label the x-axis (y-axis).
- yhf*   Specifies the y-axis high boundary.
- ylf*   Specifies the y-axis low boundary.

### *Example*

To produce a bar chart:

```
bar -r10,xa,w75 myfile
```

This outputs the bar chart described by vector `myfile`, located in region 10 of the GPS universe, with no x-axis labels. The bar width is 75% of center-to-center spacing.

## hist

### *Syntax*

```
hist [-a,b,f,g,ri,xf,xa,yf,ya,ylf,yhf] [vector . . .
```

### *Description*

The `hist` command builds a histogram. The input vector is the type produced by `bucket`, of odd rank, with odd elements being limits and even elements being bucket counts.

### *Flags*

- a*    Suppresses axes.
- b*    Plots histogram with bold weight lines (the default weight is medium).
- f*    Does not build a frame around the plot area.
- g*    Suppresses the background grid.
- ri*   Puts the histogram in GPS region *i*, where *i* is between 1 and 25 inclusive (13 by default).
- xf*
- yf*    Positions the histogram in the GPS universe with the x-origin (y-origin) at *f*.
- xa*
- ya*    Does not label the x-axis (y-axis).

- yhf Specifies the y-axis high boundary.
- ylf Specifies the y-axis low boundary.

### ***Example***

To produce a histogram:

```
hist -r5,ya myfile
```

This outputs the histogram described by vector `myfile` and locates it in region 5 of the GPS universe, with no y-axis labels.

## **label**

### ***Syntax***

```
label [-b,c,Ffile,h,p,ri,x,xu,y,yr] [GPSfile . . . ]
```

### ***Description***

The `label` command labels the axis of a GPS file.

### ***Flags***

- b Assumes that the input is a bar chart.
- c Retains lowercase letters in labels; otherwise all letters are uppercase.
- Ffile Specifies a label file. Each line of the file is taken as one label. Blank lines yield null labels. Either the GPS or the label file, but not both, can come from standard input.
- h Assumes that the input is a histogram.
- p Assumes that the input is an x-y plot. This is the default assumption.
- ri Rotates labels *i* degrees. The pivot point is the first character.
- x Labels the x-axis. This is the default action.
- xu Labels the upper x-axis (the top of the plot).
- y Labels the y-axis.
- yr Labels the right y-axis (the right side of the plot).

### *Examples*

1. To label a plot:

```
label -Flabs A.g
```

The file `A.g`, assumed to be an x-y plot, is labeled with labels from the file `labs`.

2. To label a plot from labels taken from standard input:

```
label -yr,r-45 A.g
```

The file `A.g` is labeled from the standard input. The labels are printed at 45 degrees below the horizontal.

## pie

### *Syntax*

```
pie [-b,o,p,pni,ppi,ri,v,xi,yi] [vector . . . ]
```

### *Description*

The `pie` command builds a pie chart. The input vector has a restricted format. Each input line represents a slice of the pies and has the following form:

```
[<i e f ccolor>] value [label]
```

with brackets indicating optional fields. The control field options have the following effects:

- i** The slice will not be drawn, though a space will be left for it.
- e** The slice is “exploded” or moved away from the pie.
- f** The slice is filled. The angle of fill lines depends on the color of the slice.
- ccolor** The slice is drawn in the specified color rather than the default black. Legal values are **b** (black), **r** (red), **g** (green), and **u** (blue).

The pie is drawn with the value of each slice printed inside and the label printed outside.

### *Flags*

- b** Draws pie chart with bold weight lines (the default weight is medium).
- o** Places output values around the outside of the pie.
- p** Expresses output values as a percentage of the total pie.

- p*n*** Expresses output values as a percentage, but the total of the percentages equals *i* rather than 100.
- pp*i*** Draws only *i* percent of the pie.
- r*i*** Puts the pie chart in GPS region *i*, where *i* is between 1 and 25 inclusive (13 by default).
- v** Does not output values.
- x*i***
- y*i*** Positions the pie chart in the GPS universe with x-origin (y-origin) at *i*.

### ***Example***

To draw a pie chart:

```
pie -pp80,pn80 chartfile
```

This draws the pie chart specified by `chartfile` in 80% of a circle and outputs the values as percentages of that total 80 percent.

## **plot**

### ***Syntax***

```
plot [-a,b,cstring,d,f,Ffile,g,m,ri,xf,xa,xif,xhf,xf,xni,xt,yf,ya,yif,yhf,ylf,yni,yt] [vector . . .]
```

### ***Description***

The **plot** command plots a graph. The input vectors contain the y values of an x-y graph. Values for the x-axis come from the file specified by **-F**. Axis scales are determined from the first vector plotted.

### ***Flags***

- a** Suppresses the axes.
- b** Plots the graph with bold weight lines (medium is the default weight).
- d** Does not connect plotted points (this implies **-m**).
- f** Does not build a frame around the plot area.
- Ffile** Uses the specified file for x values; otherwise the positive integers are used. You can specify this flag more than once, causing a different set of x values to be paired with each input vector. If there are more input vectors than sets of x values, the last set applies to the remaining vectors.

- g** Suppresses the background grid.
- m** Marks the plotted points.
- ri** Puts the graph in GPS region *i*, where *i* is between 1 and 25 inclusive (13 by default).
- xf**
- yf** Positions the graph in the GPS universe with the x-origin (y-origin) at *f*.
- xa**
- ya** Does not label the x-axis (y-axis).
- xa**
- ya** Does not label the x-axis (y-axis).
- xhf**
- yhf** Specifies the x-axis (y-axis) high boundary.
- xlf**
- ylf** Specifies the x-axis (y-axis) low boundary.
- xni**
- yni** Specifies the approximate number of ticks on the x-axis (y-axis).
- xt**
- yt** Omits the x-axis (y-axis) title.

### ***Examples***

1. To plot against the positive integers:

```
plot plotdata
```

2. To customize x- and y-axes:

```
plot -r5,y10,xa,Fxfile yfile
```

This plots vector *yfile* against vector *xfile*, with y-axis ticks beginning at zero, no x-axis labels being printed, and the plot being placed in region 5 of the GPS universe.

```
plot -'hilo -oy filea fileb' filea fileb
```

This plots vectors *filea* and *fileb* against the positive integers, with y-axis ticks going from the lowest to the highest values in the two vectors.

```
plot -Ffilea,Ffileb filec filed filee
```

This plots vectors *filec* against *filea*; *filed* and *filee* against *fileb*. The y-axis scale is determined from *filec*; the x-axis scale from *filea*.

**title*****Syntax***

`title [-b,c,lstring,vstring,uststring] [vector . . . ]`

***Description***

The **title** command prefixes a title to a vector or appends one to a **GPS** object.

***Flags***

- b**        Makes the **GPS** title bold.
- c**        Retains lowercase letters in the title; otherwise all letters are uppercase.
- lstring**    Uses the specified string as a **GPS** lower title.
- uststring**    Uses the specified string as a **GPS** upper title.
- vstring**    Labels a vector with the specified string.

**Related Information**

The following commands: “**ged**” on page 350, “**graphics**” on page 377, and “**spline**” on page 684.

The **gps** file in *AIX Operating System Technical Reference*.



# strip

---

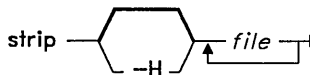
## strip

---

### Purpose

Removes symbol and line number information from a common object file.

### Syntax



OL805265

### Description

The **strip** command removes the symbol table and line number information from common object files, including archive libraries. Once you use this command, symbolic debugging of the file is difficult; therefore, you should normally run **strip** only on production modules that you have debugged and tested. Using **strip** reduces the file storage overhead required by an object file.

For each object module, **strip** removes all symbol table information. For each archive, **strip** removes the local symbol table information from each member.

You can restore a stripped symbol table to an archive or library file by using the **ar -s** command.

### Flag

**-H** Removes the object file header as well as all symbol table information.

### Files

/usr/tmp/strip\*

### Related Information

The following commands: “**ar**” on page 58, “**as**” on page 64, “**cc**” on page 112, “**dump**” on page 275, “**ld**” on page 427, “**nm**” on page 521, and “**size**” on page 665.

The **ar** and **a.out** files in *AIX Operating System Technical Reference*.

---

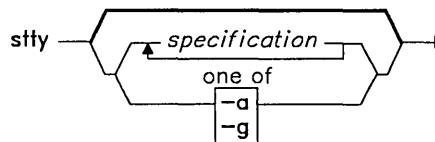
# stty

---

## Purpose

Sets, resets, or reports work station operating parameters.

## Syntax



OL805266

## Description

The `stty` command sets certain work station I/O options for the device that is the current standard input. If you run it without any *specifications*, `stty` writes to standard output information about any system adapters installed and reports the settings of certain options.

If you list any work station *specifications*, `stty` sets or resets the specified work station options.

You can find detailed information about the modes listed in the first six of the following groups in the discussion of the `termio` special facility in *AIX Operating System Technical Reference*. The last group contains options produced by combining options in the first six groups.

**Note:** The `stty` command does not make compatibility checks on any parameter combinations.

## Flags

- a Writes the current state of all option settings to standard output.
- g Writes option settings to standard output in a form usable by another `stty` command.

## Specifications

### Control Modes

The following options apply only when your work station connects to the system through an asynchronous line adapter. See *asy* in *AIX Operating System Technical Reference* for detailed information about this group.

**parenb (-parenb)** Enables (disables) parity generation and detection.

**parodd (-parodd)** Selects odd (even) parity.

**cs5 cs6 cs7 cs8** Selects character size. See *termio* in *AIX Operating System Technical Reference* for additional information on character size.

**0** Hangs up phone line immediately.

**50 75 110 134 150 300 600 1200 1800 2400 4800 9600 19200 19.2 38400 38.4 exta extb**  
Sets the work station speed to the specified number of bits per second (*exta*, 19200, and 19.2 are synonyms; *extb*, 38400, and 38.4 are synonyms). Regardless of the baud rate, the software only works with terminals that generate the ASCII character set.

**hupcl (-hupcl)**

**hup (-hup)** Hangs up (does not hang up) dial-up connection on the last close.

**cstopb (-cstopb)** Selects 2 (1) stop bits per character.

The next two options apply to all work stations, regardless of the line adapter:

**cread (-cread)** Enables (disables) the receiver.

**clocal (-clocal)** Assumes a line without (with) modem control.

### Input Modes

**ignbrk (-ignbrk)** Ignores (does not ignore) BREAK on input.

**brkint (-brkint)** Signals (does not signal) INTR on break.

**ignpar (-ignpar)** Ignores (does not ignore) parity errors.

**parmrk (-parmrk)**  
Marks (does not mark) parity errors.

**inpck (-inpck)** Enables (disables) input parity checking.

**istrip (-istrip)** Strips (does not strip) input characters to 7 bits.

**inlcr (-inlcr)** Maps (does not map) NL to CR on input.

**igncr (-igncr)** Ignores (does not ignore) CR on input.

---

|                         |                                                                                                                                                                                                                 |
|-------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>icrnl (-icrnl)</b>   | Maps (does not map) CR to NL on input.                                                                                                                                                                          |
| <b>iuclic (-iuclic)</b> | Maps (does not map) uppercase alphabetic characters to lowercase.                                                                                                                                               |
| <b>ixon (-ixon)</b>     | Enables (disables) START/STOP output control. Once START/STOP output control has been enabled, you can pause output to the work station by pressing <b>Ctrl-S</b> and resume output by pressing <b>Ctrl-Q</b> . |
| <b>ixany (-ixany)</b>   | Allows any character (only <b>Ctrl-Q</b> ) to restart output.                                                                                                                                                   |
| <b>ixoff (-ixoff)</b>   | Sends (does not send) START/STOP characters when the input queue is nearly empty/full.                                                                                                                          |

### Output Modes

|                            |                                                                                           |
|----------------------------|-------------------------------------------------------------------------------------------|
| <b>opost (-opost)</b>      | Processes output (does not process output; that is, it ignores all other output options). |
| <b>olcuc (-olcuc)</b>      | Maps (does not map) lowercase alphabetic characters to uppercase on output.               |
| <b>onlcr (-onlcr)</b>      | Maps (does not map) NL characters to CR-NL characters.                                    |
| <b>ocrnl (-ocrnl)</b>      | Maps (does not map) CR-NL characters to NL characters.                                    |
| <b>onocr (-onocr)</b>      | Does not (does) output CR characters at column zero.                                      |
| <b>onlret (-onlret)</b>    | On the terminal, NL performs (does not perform) the CR function.                          |
| <b>ofill (-ofill)</b>      | Uses fill characters (uses timing) for delays.                                            |
| <b>ofdel (-ofdel)</b>      | Uses DEL (NUL) characters for fill characters.                                            |
| <b>cr0 cr1 cr2 cr3</b>     | Selects style of delay for CR characters.                                                 |
| <b>nl0 nl1</b>             | Selects style of delay for NL characters.                                                 |
| <b>tab0 tab1 tab2 tab3</b> | Selects style of delay for horizontal tabs.                                               |
| <b>bs0 bs1</b>             | Selects style of delay for backspaces.                                                    |
| <b>ff0 ff1</b>             | Selects style of delay for form feeds.                                                    |
| <b>vt0 vt1</b>             | Selects style of delay for vertical tabs.                                                 |

**stty**

---

**Local Modes**

- isig (-isig)** Enables (disables) the checking of characters against the special control characters INTR and QUIT.
- icanon (-icanon)** Enables (disables) *canonical input* (canonical input allows input-line editing with the ERASE and KILL characters).
- xcase (-xcase)** Echoes (does not echo) uppercase characters on input, and displays uppercase characters on output with a preceding \ (backslash).
- echo (-echo)** Echoes (does not echo) every character typed.
- echoe (-echoe)** Echoes (does not echo) the ERASE character as the backspace-space-backspace string.
- Note:** This mode does not keep track of column position, so you may get unexpected results when erasing tabs, escape sequences, and the like.
- echok (-echok)** Echoes (does not echo) a NL character after a KILL character.
- lfkc (-lfkc)** Functions the same as **echok**. This is an obsolete mode.
- echonl (-echonl)** Echoes (does not echo) the NL character.
- noflsh (-noflsh)** Does not clear (does clear) buffers after INTR or QUIT.

**Control Assignments**

- control-character c* Set *control-character* to *c*, where *control-character* is **erase**, **kill**, **intr**, **quit**, **eof**, **eol**, **min**, or **time**. (Use **min** and **time** with **-icanon**.) If *c* is in the form \^*c* (backslash circumflex *c*), then its value is the corresponding CTRL character. A \^? (backslash circumflex question mark) is interpreted as DEL. A \^- (backslash circumflex minus) is interpreted as undefined.
- enhdit (-enhdit)** Enters (leaves) the enhanced line editing discipline (see the **termio** special facility in *AIX Operating System Technical Reference*).
- ascedit (-ascedit)** Enters (leaves) the ASCII keyboard mode for **dosedit**.
- line i** Sets the line discipline. *i* can be either 0 or 1. **stty line 0** is the same as **stty -enhdit**. **stty line 1** is the same as **stty enhdit**.

## Screen Length

- page** (-page) Pauses (does not pause) during output after each screen displayed. Typing any character during the pause causes output to resume. Typing a space during the pause causes output to continue uninterrupted until the next command is entered.
- length** *n* Sets screen length to *n* lines, where *n* is an integer from 1 through 255. An automatic pause in output occurs after *n* lines if **page** is enabled.

## Combination Modes

- evenp** | **parity** Enables **parenb** and **cs7**.
- oddp** Enables **parenb**, **cs7**, and **parodd**.
- parity**, **-evenp**, **-oddp**  
Disables **parenb** and sets **cs8**.
- raw** (-raw | **cooked**)  
Enables (disables) **raw** input and output (no ERASE, KILL, INTR, QUIT, EOT, or output processing).
- nl** (-nl) Unsets (sets) **icrnl** and **onlcr**. Specifying **-nl** sets **icrnl** and **onlcr** and also unsets **inlcr**, **igncr**, **ocrnl**, and **onlret**.
- lcase** (-lcase)  
**LCASE** (-LCASE)  
Sets **xcase**, **iuclc**, and **olcuc**. (Used for work stations with uppercase characters only.)
- tabs** (-tabs | **tab3**)  
Preserve tabs (expand to spaces) when printing.
- ek** Sets ERASE and KILL characters to **Ctrl-H** and **Ctrl-U**, respectively.
- sane** Resets parameters to "reasonable" values.
- term** Sets all parameters according to work station type *term*, where *term* is one of **tty33**, **tty37**, **vt05**, **tn300**, **ti700**, or **tek**.

## Terminal Mapping

- imap** *mapname* Loads **/etc/nls/termmap/mapname.in** as the terminal input map.
- omap** *mapname* Loads **/etc/nls/termmap/mapname.out** as the terminal output map.

## Examples

1. To display a short listing of your work station configuration:

```
stty
```

This lists settings that differ from the defaults.

2. To display a full listing of your work station configuration:

```
stty -a
```

3. To enable a key sequence that stops listings from scrolling off the screen:

```
stty ixon ixany
```

This sets **ixon** mode, which lets you stop runaway listings by pressing **Ctrl-S**. The **ixany** parameter allows you to resume the listing by pressing any key. The normal work station configuration includes **ixon** and **-ixany**, which allows you to stop a listing with **Ctrl-S**, but only **Ctrl-Q** will restart it.

4. To prevent all listings from scrolling off the screen:

```
stty page length 24
```

This sets page mode with a page (screen) length of 24 lines. When a listing is more than 24 lines long, the system pauses after each page. It beeps, reminding you to press any key (except the space bar) to view the next page. Press the space bar to let the rest of the listing scroll off the screen and get to the end. Paging then resumes with the next listing.

5. To reset the configuration after it has been messed up:

```
Ctrl-J stty sane echo -tabs Ctrl-J
```

Sometimes the information displayed on the screen may look strange, or the system won't respond when you press the **Enter** key. This can happen when you use **stty** with parameters that are incompatible or that do things you don't understand. It can also happen when a screen-oriented text editor ends abnormally and doesn't have a chance to reset the work station configuration.

Entering `stty sane` sets a reasonable configuration, but it may differ slightly from your normal configuration. That is why this example also includes two commonly used parameters, `echo` (erase characters as you backspace over them) and `-tabs` (expand tab characters to spaces on the display screen).

Press **Ctrl-J** before and after the command instead of **Enter**. The system usually recognizes **Ctrl-J** when the parameters that control the **Enter** key processing are messed up.

6. To save and restore the work station's configuration:

```
OLDCONFIG=`stty -g`      # save configuration
stty -echo              # do not display password
echo "Enter password: \c"
read PASSWD             # get the password
stty $OLDCONFIG         # restore configuration
```

This saves the work station's configuration, turns off echoing, reads a password, and restores the original configuration. The ` . . . ` (grave accents) in the first command tell the shell to insert the standard output of `stty -g` into the `OLDCONFIG= . . .` command. This is called **command substitution**. For more information, see "Command Substitution" on page 647.

The `stty -echo` turns off echoing, which means that the password does not appear on the screen when you type it at the keyboard. This has nothing to do with the `echo` command, which displays a message on the screen

## Related Information

The following command: "tabs" on page 729.

The `ioctl` system call and the `terminfo` and `config` files in *AIX Operating System Technical Reference*.

The discussion of `stty` and the "Overview of International Character Support" in *IBM RT PC Managing the AIX Operating System*.



**su**

---

**su**

---

## Purpose

Obtains the privileges of another user, including superuser authority.

## Syntax

```
su -c "cmdstring" user
```

The diagram illustrates the syntax of the `su` command. It shows the command `su` followed by two optional arguments in brackets. The first bracket contains `root` and `user`, representing the target user. The second bracket contains `-c "cmdstring"`, representing the command to execute as the user.

OL805267

## Description

The **su** command lets you operate with the privileges of the specified *user* (by default *root*).

If you use **su** to become the superuser, **su** sets the **PATH** variable to `/bin:/etc:/usr/bin` and changes the prompt to `#`. (Note that this **PATH** does not include the current directory.) If you are not already operating with superuser authority, **su** prompts for the password associated with *user* before granting you these privileges. Unless you enter a *command*, **su** creates a new shell that runs under *user*. This new shell is the program named in the shell field of the **passwd** file. All exported environment variables are available, unless you use the `-` flag when you call **su**.

If you need to run only one command as *user*, you can run the desired command by including it (along with any of its associated flags) on the command line as an argument to the shell `-c` flag (see “**sh**” on page 637 for a description of this flag). In this case, **su** calls the shell to run the command and exits.

Each time someone uses **su** to become the superuser, **su** writes a record in the file `/usr/adm/sulog`, (creating this file, if necessary).

To restore your normal privileges, press END OF FILE (**Ctrl-D**). This action ends the new shell, returning you to the previous shell and previous ID.

## Flag

The following flag modifies the environment of the new shell if the optional program named in the shell field of the **passwd** file is a program like **sh**.

- Creates the same environment for the new shell as the login shell of *user*. This is done by calling the new shell as a *login shell* (see “sh” on page 637), so it reads the system **profile** file and the *user’s* **\$HOME/.profile** file. The environment variables **NLLDATE** and **NLTIME** control the appearance of the date and time.

**Note:** The **TERM** and **TZ** variables are an exception. They are preserved at their current values. These variables are normally set by **init** or **getty** prior to login; hence **su** handles them differently.

## Examples

1. To obtain superuser authority:

```
su
```

This runs a subshell with the effective user ID and privileges of user **root**. The **su** command asks for a password, as if you were logging in as **root**. Now the commands you run have superuser authority. Press END OF FILE (**Ctrl-D**) to end the subshell and return to your original shell session and privileges.

2. To obtain jim’s privileges:

```
su jim
```

This runs a subshell with the effective user ID and privileges of **jim**.

3. To set up the environment as if you had logged in as **jim**:

```
su - jim
```

This runs a subshell with the effective user ID and privileges of **jim**. The **-** causes the shell variable **LOGNAME** to be set to **jim**, **HOME** to be set to the path name of **jim’s** home directory, and **jim’s** **\$HOME/.profile** shell procedure file to be run before prompting for the first shell command.

4. To run a single command with superuser authority:

```
su root -c "backup -9 -u"
```

This runs the shell command **backup -9 -u** with superuser authority (if you know the password assigned to **root**).

## Related Information

The following command: “sh” on page 637.

## sum

---

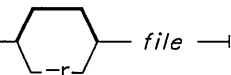
## sum

---

### Purpose

Displays the checksum and block count of a file.

### Syntax

sum  file

OL805268

### Description

The **sum** command reads *file* and calculates a 16-bit checksum for the *file* and the number of blocks in the file. The checksum and number of blocks are written to standard output. The **sum** command is generally used to determine if a file that has been copied or communicated over transmission lines is an exact copy of the original.

### Flag

**-r** Uses an alternate algorithm to compute the checksum (rigorous byte-by-byte computation rather than the default word by word computation).

### Example

To display the checksum of, and the number of blocks in `datafile`:

```
sum datafile
```

If the checksum of `datafile` is 16053 and if the file contains 3 blocks, then **sum** displays:

```
16053 3
```

### Related Information

The following command: “**wc**” on page 846.

## **sync**

---

### **Purpose**

Updates the superblock and writes buffered files to the fixed disk.

### **Syntax**

`sync` —|

OL805221

### **Description**

The **sync** command runs the **sync** system primitive. If you have to stop the system, you must run **sync** to ensure file system integrity. **sync** writes all unwritten system buffers to disk. This includes modified superblocks, modified i-nodes, delayed block I/O, and read-write mapped files.

**Note:** The writing, although scheduled, is not necessarily complete upon return from the **sync** system call.

### **Related Information**

The **sync** system call in *AIX Operating System Technical Reference*.

## tab

---

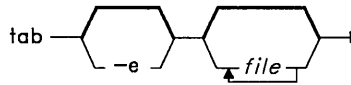
## tab, untab

---

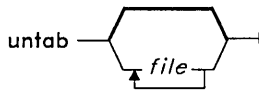
### Purpose

Changes space characters into tabs.

### Syntax



OL805069



OL805065

### Description

The **tab** command reads *files* (standard input by default), replaces spaces in the input with tab characters wherever it can eliminate one or more spaces. It writes the resulting file back to *file* or, if the input was standard input, to standard output. **tab** assumes that the tab stops are set every eight columns starting with column nine.

The **untab** command reads *files* or standard input, replaces tabs in the input with space characters and writes back to the original file or to standard output.

### Flag

- e** Replaces only those spaces at the beginning of a line up to the first nonspace character.

### Related Information

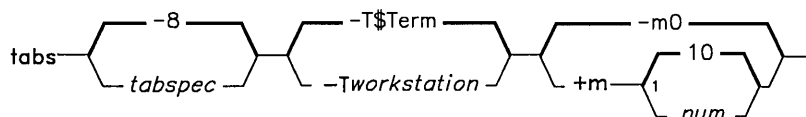
The following command: “**newform**” on page 507.

# tabs

## Purpose

Sets tab stops on work stations.

## Syntax



<sup>1</sup> Do not put a space between these items.

OL805381

## Description

The `tabs` command clears up to 20 previous tabs and sets up to 40 tabs on the work station according to the supplied *tabspec*. *tabspec* can be either a flag indicating an available code or column numbers. The available codes cover formats required by most structured programming languages.

When you use the `tabs` command, always see the leftmost column number as 1, even if your work station refers to it as zero (0).

If you do not specify a *tabspec*, the default value is `-8`.

## Tabspecs

- a Sets the tabs to 1, 10, 16, 36, and 72 (IBM S/370 Assembler first format)
- a2 Sets the tabs to 1, 10, 16, 40, and 72 (IBM S/370 Assembler second format)
- c Sets the tabs to 1, 8, 12, 16, 20, and 55 (COBOL normal format)
- c2 Sets the tabs to 1, 6, 10, 14, and 49 (COBOL compact format, columns 1-6 omitted).  
With this code, the first column position corresponds to card column 7. One space gets you to column 8, and a tab reaches column 12. Files using this code should include a format specification of:

```
<:t-c2 m6 s66 d:>
```

## tabs

---

For an explanation of format specifications, see the **fspec** file in *AIX Operating System Technical Reference*.

- c3** Sets the tabs to 1, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62, and 67 (COBOL compact format with more tabs than **-c2**). This is the recommended format for COBOL. Files using this code should include a format specification of:

```
<:t-c3 m6 s66 d:>
```

- f** Sets the tabs to 1, 7, 11, 15, 19, and 23 (FORTRAN).
- p** Sets the tabs to 1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, and 61 (PL/I).
- s** Sets the tabs to 1, 10, and 55 (SNOBOL).
- u** Sets the tabs to 1, 12, 20, and 44.

In addition to the preset formats, three other types of *tabspecs* are available:

- num** Sets regularly repeating tabs at every *num*th column. (-8 is the standard AIX tab setting and the one required for use with the **nroff -h** flag.) Another special case is **-0**, which implies no tabs at all.
- num[,num] . . .** Sets tabs at the named column numbers (a comma-separated list in ascending order). You may specify up to 40 numbers. If any number except the first has a plus sign prefix, the prefixed number is added to the previous number for the next setting. Thus, the tab lists 1,10,20,30 and 1,10,+10,+10 provide the same tab settings.
- filep** Reads the first line of the named *filep* for a format specification. If it finds one, it sets tabs the same way. If it does not find a format specification, it sets tabs to the system default (-8). Use this *tabspec* to make sure that a file has the same tab settings as those in a file already correctly formatted.

## Flags

**Note:** If the same flag occurs more than once, only the last one takes effect.

- Tworkstation** Identifies the work station so that **tabs** can set tabs and margins correctly. *workstation* is one of the work stations listed under the **greek** command. If you do not provide a **-T** flag, **tabs** uses the shell variable **\$TERM**. If no *workstation* can be found, **tabs** tries a general value that works for most work stations.
- +mnum**
- +mnum** Moves all tabs to the right *num* columns, and makes column *num1* the left margin. If **m** is given without a value, 10 is assumed. The leftmost margin on most work stations is defined by **m0**.

## Related Information

The following commands: “**greek**” on page 379, “**nroff**” on page 525, and “**troff**” on page 526.

The discussion of **term** and **environ** in *AIX Operating System Technical Reference*.



# tail

---

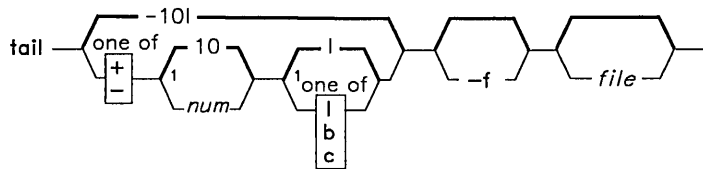
# tail

---

## Purpose

Writes a file to standard output, beginning at a specified point.

## Syntax



OL805303

<sup>1</sup> Do not put a blank between these items.

OL805308

## Description

The `tail` command writes the named *file* (standard input by default) to standard output, beginning at a point you specify. It begins reading at `+ [num]` lines from the beginning of *file* or `- [num]` lines from the end of *file*. The default *num* is 10. *num* is counted in units of lines, blocks, or characters, according to the subflag appearing after *num* (see the following flags).

## Flags

**-f** Does not end after it copies the line of the input file if the input file is not read from a pipe, but enters an endless loop in which it sleeps for a second and then attempts to read and copy further records from the input file. Thus, it can be used to monitor the growth of a file being written by another process.

`+ [num]l`  
`+ [num]b`  
`+ [num]c`

Begins reading *num* lines (`l`, the default), blocks (`b`), or characters (`c`) from the beginning of the input.

---

`-[num]l`  
`-[num]b`  
`-[num]c` Begins reading *num* lines (**l**, the default), blocks (**b**), or characters (**c**) from the end of the input.

## Examples

1. To display the last 10 lines of a file:

```
tail notes
```

2. To specify how far from the end to start:

```
tail -20 notes
```

This displays the last 20 lines of *notes*.

3. To specify how far from the beginning to start:

```
tail +200c notes | pg
```

This displays *notes* a page at a time starting with the 200th character from the beginning.

4. To follow the growth of a file:

```
tail -1 -f accounts
```

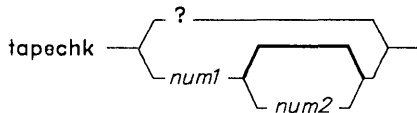
This displays the last line of *accounts*. Once a second, **tail** displays any lines that have been added to the file. This continues until stopped by pressing INTERRUPT (**Alt-Pause**).

## Related Information

The following command: “**dd**” on page 228 and “**pg**” on page 553.

**tapechk****tapechk****Purpose**

Performs consistency checking of the streaming tape device.

**Syntax**

OL805445

**Description**

The **tapechk** command performs rudimentary consistency checking on an attached streaming tape device. Some hardware malfunctions with a streaming tape drive can be detected by simply reading a tape. **tapechk** provides a way to perform tape reads on the file level.

Since the streaming tape drive cannot backspace over physical data blocks or files, **tapechk** rewinds the tape to its starting position prior to each check. You can specify numeric arguments to control the number of files checked or skipped. If you do not specify any arguments, **tapechk** rewinds the tape and checks only the first physical block.

Although you can use **tapechk** on any streaming tape cartridge, it is primarily designed for checking tapes written by the **backup** command.

**Flags**

- num1* Checks data for the next *num1* files.
  - num2* Skips the next *num2* files from the beginning of the tape.
  - ? Explains the format of the **tapechk** command.
- Note:** If you specify this argument, it must be the first argument.

---

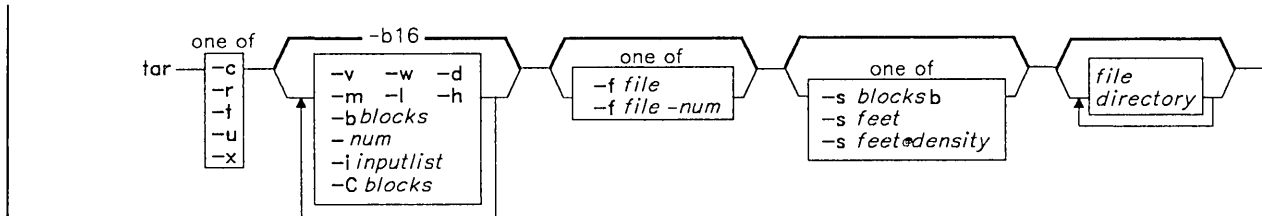
# tar

---

## Purpose

Manipulates tape archives.

## Syntax



OL805423

## Description

The **tar** command writes *files* to or retrieves *files* from archival storage. **tar** normally looks for archives on a magnetic tape, but you can specify other archival files with the **-f** flag. File names must not be longer than 100 characters and must not contain blanks (**tar** ignores all characters following the first blank.)

When writing to an archive, **tar** uses a temporary file (**/tmp/tar\***) and maintains in memory a table of files with several links. You will receive an error message if **tar** cannot create the temporary file, or if there is not enough memory available to hold the link tables.

Backspacing is not supported on a *raw magnetic tape* device, so the **-u** and **-r** flags will rewind the tape, and then open and read it again. Backspacing is supported and used on ordinary files and block special files. Records are always one block long on block magnetic tape, but they are typically less than half as dense as blocked records on raw magnetic tape (because of inter-record gaps). Therefore, although a blocked raw tape must be read twice, the total amount of tape motion is less than it is when reading one-block records from a block magnetic tape once.

**Note:** There is no way to ask for any occurrence of a file other than the last, and there is no recovery from tape errors.

## Flags

You must supply one of the following five function flags to control the actions of **tar**:

- c** Creates a new archive from the named *files*. Writing begins at the beginning of the tape.
- r** Writes *files* at the end of the tape.
- t** Lists the *files* in the order in which they appear on the tape. Files may appear more than once on a tape.
- u** Adds *files* to the tape only if they are not already there or if they have been modified since last written onto the tape.
- x** Extracts *files* from the tape. If a *file* specifies a directory, **tar** extracts all files on the tape in that directory. If you do not specify a *file*, **tar** extracts all files on the tape. If multiple copies of the same file are on the tape, **tar** extracts only the last one and overwrites all earlier ones. If you have superuser authority (see “**su**” on page 724), **tar** creates all files and directories with the same user and group IDs as on the tape. If you do not have superuser authority, the files and directories have your user and group IDs.

The other optional flags to **tar** are listed below. In all cases, a directory parameter refers to all the files and subdirectories, recursively, within that directory. Flags without corresponding parameters may appear separately or be grouped together. Flags that take parameters may have them adjacent to the flag letter or as the entire following argument.

**-bblocks** Specifies the number of 512-byte blocks per tape record. The default is 16, which is appropriate for tape records. Due to the size of inter-record gaps, tapes written with large blocking factors can hold much more data than tapes with only one block per record.

The block size is determined automatically when tapes are read (function flags **-x** or **-t**). When archives are updated with the **-u** and **-r** functions, the existing record size is used. **tar** will write archives using the specified *blocks* value only when creating new archives under the **-c** function.

For output to ordinary files with the **-f** flag, you can save disk space by using a blocking factor that matches the size of disk blocks (for example, **-b4** for 2048-byte disk blocks). Ordinary files must be read using the same blocking factor used when they were created.

**-Cblocks** Allows **tar** to use very large clusters of blocks when it deals with streaming tape archives. Note, however, that on input, **tar** cannot automatically determine the block size of tapes with very long block sizes created with this flag. In the absence of a **-Cnum** argument, the largest block size that **tar** can automatically determine is 20 blocks.

- 
- d** Makes separate tape entries for directories, blocks and character special files, and FIFOs. Normally, **tar** writes only ordinary files to tape, and extracts only ordinary files and the directories required to contain them as determined by the path names on the tape. When writing to tape with the **-d** flag, **tar** makes it possible to preserve the directory permission codes and to restore empty directories, special files, and FIFOs with the **-x** flag.
- Note:** Although anyone can archive special files, only a user with superuser authority can extract them from the tape.
- ffile[-num]** Uses *file* as the archive to be read or written. Without this flag, **tar** uses a system-dependent default file name of the form */dev/rmt?*. If the *file* given is **-** (minus), **tar** writes to standard output or reads from standard input. If you write to standard output, the **-c** function flag must be used.
- If you specify *num*, **tar** provides automatic spillover from one file or tape unit to another. For example, **-f/dev/rmt0-2** writes or reads */dev/rmt0*, followed by */dev/rmt1*, and then */dev/rmt2* before requesting that additional volumes be mounted. This feature allows the operator of a system with multiple tape drives to use multi-tape archives without having to change tapes.
- h** Ignores header checksum errors. **tar** writes a file header containing a checksum for each file on the tape. Without this flag, when it reads a tape, it verifies the contents of the header blocks by recomputing the checksum, and aborts with a directory checksum error when a mismatch occurs. With this flag, **tar** logs the error and then scans forward on the tape until it finds a valid header block. This permits restoring files from later volumes of a multi-volume archive without reading earlier volumes.
- iinputlist** Writes the files named in the file *inputlist* to the archive. *inputlist* contains one file name per line. Files from *inputlist* are not treated recursively. If you include the name of a directory in *inputlist*, **tar** does not write that directory's subdirectories to the tape, only that directory's files. If *files* are listed on the command line, the contents of *inputlist* are included after **tar** has written all the *files* and their subdirectories to the archive.
- l** Writes error messages to standard output if **tar** cannot resolve all of the links to the files archived. (If you do not specify this flag, you will not get these messages.)
- m** Uses the time of extraction as the modification time, even if invoked by a user with superuser authority. Usually, when **tar** is invoked by a user with superuser authority, it preserves the modification time of extracted files.

## tar

---

- s** *blocks***b**
- s** *feet*
- s** *feet @density*      Specifies the number of 512-byte *blocks* per volume (first format), independent of the tape blocking factor. You can also specify the size of the tape in feet by using the second form, and **tar** assumes a default *density*. The third form allows you to specify both tape length and density. Feet are assumed to be 11 inches long to be conservative. This flag lets you deal more easily with multi-volume tape archives, where **tar** must be able to determine how many blocks fit on each volume.
  
- v**      Lists the name of each file as it is processed. With the **-t** function flag, **-v** gives more information about the tape entries, including file sizes, times of last modification, UID, and GID, and permissions.
  
- w**      Displays the action to be taken followed by the file name, then wait for user confirmation. If the response begins with *y* or *Y*, the action is performed; otherwise, the file is ignored.
  
- num**      Uses **/dev/rmtnum** instead of the default. For example, **-2** is the same as **-f/dev/rmt2**. In AIX systems with multi-density tape drives, this flag allows selecting a particular density. The default unit is system dependent and is chosen to match the default density, as described under the **-s** flag.

## Files

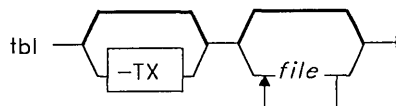
*/dev/rmt?*  
*/tmp/tar\**  
*/bin/find*  
*/bin/sort*

## tbl

## Purpose

Formats tables for the **nroff** and **troff** commands.

## Syntax



OL805222

## Description

The **tbl** command is a preprocessor that formats tables for **nroff** and **troff**. It reads the specified *files* or, if you do not specify any file names or you specify a - (minus) as a file name, it reads standard input. The input is copied unchanged to standard output, except for text between lines containing **.TS** and **.TE**. This text describes tables, and is reformatted by **tbl**. The **.TS** and **.TE** lines are not altered by **tbl**. For more detailed information on how to format text for **tbl**, see *Text Formatting Guide*.

**Note:** When **tbl** is used with **eqn** or **neqn**, **tbl** should come first to minimize the volume of data passed through pipelines.

At the start of **tbl** text, you should include a line containing **.TS**. You can follow this with a line containing global options. The available global options are:

|                     |                                                                                         |
|---------------------|-----------------------------------------------------------------------------------------|
| <b>center</b>       | Centers the table (the default is left-adjusted).                                       |
| <b>expand</b>       | Makes the table as wide as the current line length.                                     |
| <b>box</b>          | Encloses the table in a box.                                                            |
| <b>doublebox</b>    | Encloses the table in a double box.                                                     |
| <b>allbox</b>       | Encloses each item of the table in a box.                                               |
| <b>tab (x)</b>      | Uses the character <i>x</i> instead of a tab to separate items in a line of input data. |
| <b>linesize (n)</b> | Sets lines and rules for boxes in point size <i>n</i> .                                 |
| <b>delim (x,y)</b>  | Sets <i>x</i> and <i>y</i> as the <b>eqn</b> and <b>neqn</b> text delimiters.           |

End the list of global options with a ; (semicolon).

After the global options, enter lines describing the format of each row in the table. Each format line (except the last) describes one row of the table. The last one describes all remaining rows of the table. This must end with a period to indicate that it is the end of



the format specification. Each column of the table is described by a single keyletter. The available keyletters are:

- c** Centers the item in the column.
- r** Right-adjusts the item in the column.
- l** Left-adjusts the item in the column.
- n** Adjusts the numerical items in the column to line up at the decimal point or right-adjusts them if there are no decimal points.
- s** Allows the previous item on the left to spill over into this column if the item is too wide for its column.
- a** Centers the longest line in this column and then left-adjusts all other lines in it with respect to the centered line.
- ^** Allows the item above to spill over into this column if the item is too large.
- \_** Replaces this entry with a horizontal line.
- =** Replaces this entry with a double horizontal line.

After the keyletter, you can enter specifiers that determine where vertical lines appear between columns, column width, inter-column spacing, and the font and point size of the item. See “Column and Item Specifiers” on page 745 for legal specifiers.

The format lines are followed by lines containing data for the table. The last line consists of **.TE**. Within the data lines, data items are separated by tab characters, unless the global option, **delim** is used.

If a data line consists of only **\_** (underscore) or **=** (equal sign), a single or double line is drawn across the table at that point. If an entry in a data line consists of only **\_** or **=**, then that item is replaced by a single or double line.

| Specifier            | Meaning                   | Specifier            | Meaning                        |
|----------------------|---------------------------|----------------------|--------------------------------|
| <b>e</b> or <b>E</b> | Equal width columns.      | <b>w</b> or <b>W</b> | Minimum width column.          |
| <b>f</b> or <b>F</b> | Font change.              | <b>z</b> or <b>Z</b> | Zero width column.             |
| <i>nnn</i>           | Column separation.        | <i>.xx</i>           | Included <b>troff</b> request. |
| <b>p</b> or <b>P</b> | Point size change.        |                      | Vertical line.                 |
| <b>s</b> or <b>S</b> | Spanned item.             |                      | Double vertical line.          |
| <b>t</b> or <b>T</b> | Vertical spanning at top. | \^                   | Vertical span.                 |
| <b>T{ . . . T }</b>  | Text block.               | \_                   | Short horizontal line.         |
| <b>u</b> or <b>U</b> | Staggered columns.        | \R <i>x</i>          | Repeat character.              |
| <b>v</b> or <b>V</b> | Vertical spacing change.  |                      |                                |

Figure 6. tbl Column and Item Specifiers

## Flag

- TX** Uses only full vertical line motions, making the output suitable for line printers and other devices that do not have partial vertical line motions.

## Related Information

The following commands: “**cw, checkcw**” on page 213, “**eqn, neqn, checkeq**” on page 300, “**mm, checkmm**” on page 492, “**mmt, checkmm**” on page 495, “**nroff**” on page 525, and “**troff**” on page 526.

The **mm** and **mv** miscellaneous facilities in *AIX Operating System Technical Reference*.

The discussion of **tbl** in *Text Formatting Guide*.

**tc**

---

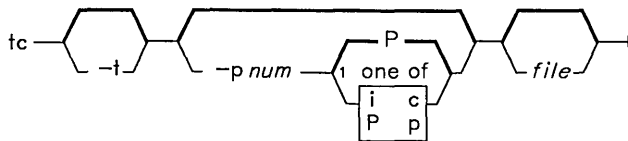
**tc**

---

## Purpose

Simulates phototypesetter output for a Tektronix 4014 work station.

## Syntax



OL805271

---

<sup>1</sup> Do not put a blank between these items.

OL805308

## Description

The `tc` command interprets its input, either a *file* or standard input, as a **troff** document. It then simulates the typesetter output for a Tektronix 4014 work station with ASCII and APL character sets and writes the results to standard output (usually the work station display). The 16 typesetter sizes are mapped into the 4014's four sizes; the entire **troff** character set is drawn using the 4014's character generator, with overstruck combinations where necessary.

At the end of each page, `tc` waits for a new-line character from the keyboard before continuing to the next page. While it is waiting, the command `e` suppresses the screen erase before the next page. `!AIX-cmd` sends `AIX-cmd` to the shell.

## Flags

- `-pnum letter` Sets page length to *num* and scale to *letter*. *letter* may include the scale factors **p** (points), **i** (inches), **c** (centimeters), and **P** (picas). The default is picas. Do not put a space between *num* and *letter*.
- `-t` Does not wait between pages (use in a pipeline).

There are no font distinctions in the display.

## Example

To use `tc` in a pipeline with `troff`:

```
troff -t chapter1 | tc
```

## Related Information

The following commands: “`sh`” on page 637, “`tplot`” on page 762, “`troff`” on page 526, and “`4014`” on page 865.

# tctl

---

# tctl

---

## Purpose

Gives commands to streaming tape.

## Syntax

```
tctl -f$TAPE -ftapename subcmd 1 count
```

OL805397

## Description

The **tctl** command gives subcommands to a streaming tape device. If you do not specify the **-f** flag with *tapename*, the environment variable **TAPE** is used. If the environment variable does not exist, **tctl** uses the device **/dev/rmt4**. The *tapename* parameter must be a raw (not block) tape device. You can specify more than one operation with *count*.

## Subcommands

- eof**
- weof** Writes *count* end-of-file markers at the current position on the tape.
- fsf** Moves the tape forward *count* files.
- fsr** Moves the tape forward *count* records.
- rewind** Rewinds the tape. The *count* parameter is ignored.  
**Note:** It is sometimes necessary to issue a **reset** before issuing a **rewind** subcommand.
- offline**
- rewoffl**
- reset** Places the tape drive off-line. The *count* parameter is ignored.
- erase** Erases all contents on the tape and rewinds it.
- retension** Moves the tape to the beginning, the end, and back to the beginning of the tape. If you have excessive read errors during a restore operation, you should run the **retension** subcommand. If the tape has been exposed to environmental extremes, you should run the **retension** subcommand before the save operation.

- ras1** Performs a checksum on the tape drive.
- ras2** Checks the capstan speed, verifies the operations of the BOT, EOT, and SAFE sensors, and writes a worst case pattern on the tape and attempts to verify the pattern.

## Files

/dev/rmt?? The raw streaming tape interface.

## Related Information

The following command: “**dd**” on page 228.

The **ioctl** system call and the **tape** and **environ** files in *AIX Operating System Technical Reference*.

# tee

---

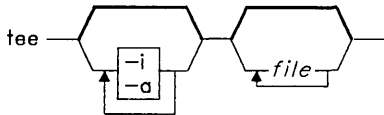
## tee

---

### Purpose

Displays the output of a program and copies it into a file.

### Syntax



OL805272

### Description

The **tee** command reads standard input and writes the output of a program to standard output and copies it into *file* at the same time.

### Flags

- a Adds the output to the end of *file* instead of writing over it.
- i Ignores interrupts.

**Note:** If you specify both flags, each must appear separately on the command line, preceded by a - (minus).

### Examples

1. To view and save the output from a command at the same time:

```
lint program.c | tee program.lint
```

This displays the standard output of the command `lint program.c` at the workstation, and at the same time saves a copy of it in the file `program.lint`. If `program.lint` already exists, it is deleted and replaced.

2. To display and append to a file:

```
lint program.c | tee -a program.lint
```

This displays the standard output of `lint program.c` at the work station and at the same time appends a copy of it to the end of `program.lint`. If the file `program.lint` does not exist, it is created.



## termdef

---

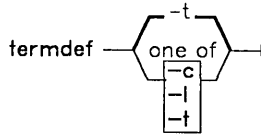
## termdef

---

### Purpose

Queries terminal characteristics.

### Syntax



OL805454

### Description

The **termdef** command identifies the current display type, the active lines setting, or the current columns setting, thus simplifying the task of resetting the lines and columns when you switch fonts or of resetting the **\$TERM** environment variable when you switch displays. The **terminfo** file defines the default number of lines and columns for each display, but the lines and columns can change depending upon which font is currently active. In addition, the **\$TERM** environment variable does not automatically reflect the display currently being used. If you are using a display other than the **ibm5151**, you must explicitly reset this variable to access the **terminfo** correctly.

### Flags

- c Returns the current column value.
- l Returns the current lines value.
- t Returns the name of the current display (this is the default action).

### Example

To set environment variables according to the values of the currently active font and display, add the following lines to the **/etc/rc** file:

```
TERM=`termdef`  
COLUMNS=`termdef -c`  
LINES=`termdef -l`  
export TERM LINES COLUMNS
```

## **Related Information**

The following command: “**display**” on page 258.

The **terminfo** file and the **hft** special file in *AIX Operating System Technical Reference*.

# test

---

## test

---

### Purpose

Evaluates conditional expressions.

### Syntax

```
test — expression —  
  
[ — expression — ] —
```

OL805273

### Description

The **test** command evaluates *expression* and, if its value is true, returns a zero (true) exit value; otherwise it returns a nonzero (false) exit value; **test** also returns a nonzero exit value if there are no parameters.

**Note:** In the second form of the command, that is the one that uses square brackets ([ ]), rather than the word **test**, the brackets must be surrounded by blanks.

### Functions

All the functions and operators are separate parameters to **test**. The following functions are used to construct *expression*:

|                       |                                                             |
|-----------------------|-------------------------------------------------------------|
| <b>-r</b> <i>file</i> | True if <i>file</i> exists and has read permission.         |
| <b>-w</b> <i>file</i> | True if <i>file</i> exists and has write permission.        |
| <b>-x</b> <i>file</i> | True if <i>file</i> exists and has execute permission.      |
| <b>-f</b> <i>file</i> | True if <i>file</i> exists and is a regular file.           |
| <b>-d</b> <i>file</i> | True if <i>file</i> exists and is a directory.              |
| <b>-c</b> <i>file</i> | True if <i>file</i> exists and is a character special file. |
| <b>-b</b> <i>file</i> | True if <i>file</i> exists and is a block special file.     |
| <b>-p</b> <i>file</i> | True if <i>file</i> exists and is a named pipe (FIFO).      |
| <b>-u</b> <i>file</i> | True if <i>file</i> exists and its set-user-ID bit is set.  |

---

|                                |                                                                                                                                                                                                      |
|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>-g</b> <i>file</i>          | True if <i>file</i> exists and its set-group-ID bit is set.                                                                                                                                          |
| <b>-k</b> <i>file</i>          | True if <i>file</i> exists and its <b>sticky bit</b> is set.                                                                                                                                         |
| <b>-s</b> <i>file</i>          | True if <i>file</i> exists and has a size greater than zero.                                                                                                                                         |
| <b>-t</b> [ <i>filedescr</i> ] | True if the open file with file descriptor number <i>filedescr</i> (1 by default) is associated with a work station device.                                                                          |
| <b>-z</b> <i>s1</i>            | True if the length of string <i>s1</i> is zero.                                                                                                                                                      |
| <b>-n</b> <i>s1</i>            | True if the length of the string <i>s1</i> is nonzero.                                                                                                                                               |
| <i>s1</i> = <i>s2</i>          | True if strings <i>s1</i> and <i>s2</i> are identical.                                                                                                                                               |
| <i>s1</i> != <i>s2</i>         | True if strings <i>s1</i> and <i>s2</i> are not identical.                                                                                                                                           |
| <i>s1</i>                      | True if <i>s1</i> is not the null string.                                                                                                                                                            |
| <i>n1</i> <b>-eq</b> <i>n2</i> | True if the integers <i>n1</i> and <i>n2</i> are algebraically equal. Any of the comparisons <b>-ne</b> , <b>-gt</b> , <b>-ge</b> , <b>-lt</b> , and <b>-le</b> can be used in place of <b>-eq</b> . |

These functions can be combined with the following operators:

|                         |                                                                        |
|-------------------------|------------------------------------------------------------------------|
| <b>!</b>                | Unary negation operator.                                               |
| <b>-a</b>               | Binary AND operator.                                                   |
| <b>-o</b>               | Binary OR operator ( <b>-a</b> has higher precedence than <b>-o</b> ). |
| <b>\( expression \)</b> | Parentheses for grouping.                                              |

## Examples

1. To test whether a file exists and is not empty:

```
if test ! -s "$1"
then
    echo $1 does not exist or is empty.
fi
```

If the file specified by the first positional parameter to the shell procedure does not exist, this displays an error message. If \$1 exists, it displays nothing. Note that there must be a space between **-s** and the file name.

The double quotes around \$1 ensure that the test will work properly even if the value of \$1 is the empty string. If the double quotes are omitted and \$1 is the empty string, **test** displays the error message `test: parameter expected`.

## test

---

2. To do a complex comparison:

```
if [ $# -lt 2 -o ! -s "$1" ]
then
    exit
fi
```

If the shell procedure was given fewer than two positional parameters or the file specified by \$1 does not exist, then this exits the shell procedure. The special shell variable \$# represents the number of positional parameters entered on the command line that started this shell procedure. For more details, see “Shell Variables and Command-Line Substitutions” on page 641.

## Related Information

The following commands: “**find**” on page 326 and “**sh**” on page 637.

---

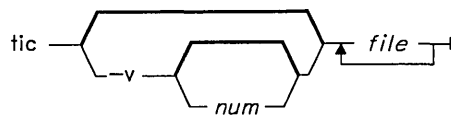
## tic

---

### Purpose

Translates **terminfo** files from source to compiled format.

### Syntax



OL805340

### Description

The **tic** command translates **terminfo** files from the source format into the compiled format. **tic** places the results in the directory `/usr/lib/terminfo`. If the environment variable **TERMINFO** is set, the results are placed there instead of in `/usr/lib/terminfo`.

The **tic** command compiles all terminfo descriptions in *files*. When **tic** finds a **use =** field, it searches first the current file, then the master file, `./terminfo.src`.

The total compiled entries cannot exceed 4096 bytes and the name field cannot exceed 128 bytes.

### Flag

**-vnum** Writes trace information on the progress of **tic**. *num* is an integer that increases the level of the verbosity.

### Files

`/usr/lib/terminfo/?/*` Compiled terminal capability data base.

### Related Information

The **curses** subroutine and the **terminfo** file in *AIX Operating System Technical Reference*.

# time

---

## time

---

### Purpose

Times the execution of a command.

### Syntax

```
time — command —
```

OL805274

### Description

The **time** command times the execution of the named *command*. **time** writes to standard error the elapsed time of the command, the system time used, and the execution time, in seconds.

### Examples

1. To measure the time required to run a program:

```
time a.out
```

This runs the program **a.out** and writes to the standard error output the amount of real, system, and user time that it uses:

```
real      10.5
user       0.3
sys       3.6
```

2. To save a record of the **time** information in a file:

```
time a.out 2> a.time
```

### Related Information

The following command: “**timex**” on page 755.

The **times** system call in *AIX Operating System Technical Reference*.

---

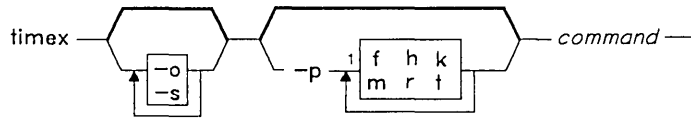
# timex

---

## Purpose

Times a command, and reports process data and system activity.

## Syntax



<sup>1</sup>Do not put a blank between these items.

OL805275

## Description

The **timex** command reports, in seconds, the elapsed time, user time, and system execution time for *command*. With flags specified, **timex** can list or summarize process accounting data for *command* and all of its children, and report total system activity during the execution interval. The output of the **timex** command is written to standard error.

**Note:** **timex** only reports on local commands.

The **timex** commands uses the accounting file `/usr/adm/pacct` to select process records associated with *command*. It also includes background processes having the same user ID, work station ID, and execution time window.

## Flags

- o Reports the total number of blocks read or written and total characters transferred by *command* and all its children.
- p Lists process accounting records for *command* and all its children. The **f**, **h**, **k**, **m**, **r**, and **t** arguments modify the data items reported and are defined in the **acctcom** command (see page 38.) The number of blocks read or written and the number of characters transferred are always reported.
- s Reports total system activity that occurred during the execution of *command*. All the data items listed in **sar** are reported (see page 614).



## **Related Information**

The following commands: “**acctcom**” on page 38 and “**sar**” on page 614.

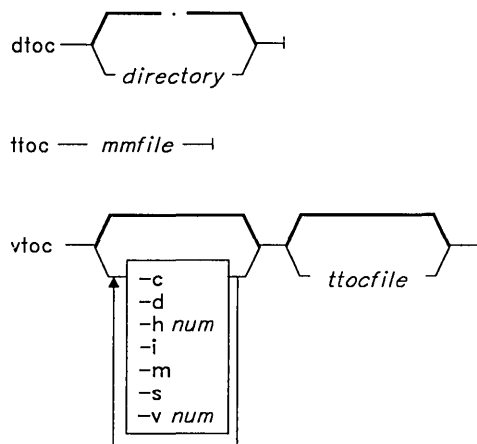
---

**toc**


---

**Purpose**

Provides graphical table of contents routines.

**Syntax**


OL777076

**Description**

All of the commands listed below reside in `/usr/bin/graf` (see “**graphics**” on page 377).

**dtoc**

The **dtoc** command makes a textual table of contents, **TTOC**, of all subdirectories beginning at *directory* (by default the current directory.). The list has one entry per directory. The entry fields from left to right are level number, directory name, and the number of ordinary readable files in the directory. **dtoc** is useful in making a visual display of all or parts of a file system. The following will make a visual display of all the readable directories under the root directory (`/`):

```
dtoc / | vtoc | td
```

### ttoc

Output is the table of contents generated by the .TC macro of the **mm** command translated to **TTOC** format. The input is assumed to be a **mm** file that uses the .H family of macros for section headers. If no *file* is given, the standard input is assumed.

### vtoc

The **vtoc** command produces a **GPS** describing a hierarchy chart from a **TTOC**. The output drawing consists of boxes containing text connected in a tree structure. If no *file* is given, the standard input is assumed. Each **TTOC** entry describes one box and has the form:

*id*[*line-weight,line-style*]"*text*"[*mark*]

where:

*id* is an alternating sequence of numbers and dots. The *id* specifies the position of the entry in the hierarchy. The *id* **0.** is the root of the tree.

*line-weight* is either:

**n**, normal-weight; or  
**m**, medium-weight; or  
**b**, bold-weight.

*line-style* is either:

**so**, solid-line;  
**do**, dotted-line;  
**dd**, dot-dash line;  
**da**, dashed-line; or  
**ld**, long-dashed

*text* is a character string surrounded by quotes. The characters between the quotes become the contents of the box. To include a quote within a box it must be escaped (\").

*mark* is a character string (surrounded by quotes if it contains spaces), with included dots being escaped. The string is put above the top right corner of the box. To include either a quote or a dot within a *mark* it must be escaped.

Entry example:

1.1b,da"ABD" DEF

Entries may span more than one line by escaping the new-line (**\new-line**).

Comments are surrounded by the **/\*,\*/** pair. They may appear anywhere in a **TTOC**.

### ***Flags***

- c** Uses text as entered, (default is all upper case).
- d** Connects the boxes with diagonal lines.
- hnum** Sets horizontal interbox space to *num*% of box width.
- i** Suppresses the box *id*.
- m** Suppresses the box *mark*.
- s** Do not compact boxes horizontally.
- vnum** Vertical interbox space is *num*% of box height.

### **Related Information**

The following command: “**graphics**” on page 377.

The **gps** file in *AIX Operating System Technical Reference*.

# touch

---

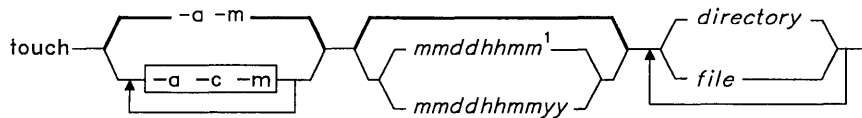
## touch

---

### Purpose

Updates the access and modification times of a file.

### Syntax



<sup>1</sup>The current year is the default year.

OL805276

### Description

The **touch** command updates the access and modification times of each *file* or *directory* named to the one specified on the command line. If you do not specify a time, **touch** uses the current time. If you specify a file that does not exist, **touch** creates a file with that name unless you request otherwise with the **-c** flag.

The environment variables **NLDATE** and **NLTIME**, if defined, specify the order of month and day in the date specification and of hour and minute in the time specification. Otherwise, these orders default to *mmdd* and *hhmm*.

The return code from **touch** is the number of files for which the times could not be successfully modified (including files that did not exist and were not created).

### Flags

- a** Changes only the access time.
- c** Does not create the file if it does not already exist.
- m** Changes only the modification time.

## Examples

1. To update the access and modification times of a file:

```
touch program.c
```

This sets the last access and last modification times of `program.c` to the current date and time. If `program.c` does not exist, **touch** creates an empty file with that name.

2. To avoid creating a new file:

```
touch -c program.c
```

3. To update only the modification time:

```
touch -m *.o
```

This updates only the last modification times of the files in the current directory that end with `.o`. **touch** is often used in this way to alter the results of the **make** command.

4. To explicitly set the access and modification times:

```
touch -c 02171425 program.c
```

This sets the access and modification dates to 14:25 (2:25 p.m.) February 17 of the current year.

## Related Information

The following command: “**date**” on page 219.

The **utime** system call in *AIX Operating System Technical Reference*.

The “Overview of International Character Support” in *Managing the AIX Operating System*.

# tplot

---

## tplot

---

### Purpose

Produces plotting instructions for a particular work station.

### Syntax

```
tplot -T$TERM -Tworkstation file
```

OL805277

### Description

The **tplot** command reads plotting instructions from standard input or from *file*, if specified. (For more information about plotting instructions, see the **plot** file format *AIX Operating System Technical Reference*). **tplot** writes instructions suitable for the specified *workstation* to standard output. If *workstation* is not specified, the environment variable **TERM** is used. (For more information about environment variables, see the **environ** file in *AIX Operating System Technical Reference*).

### Flag

**-Tworkstation** Uses the plotting instructions for *workstation*. The known *workstation* is:

|           |                         |
|-----------|-------------------------|
| <b>lp</b> | IBM PC graphics printer |
|-----------|-------------------------|

### Files

/usr/lib/tcolor  
/usr/lib/tprint

### Related Information

The following commands: “**graph**” on page 375 and “**splp**” on page 687.

The **plot** subroutine and the **plot** file in *AIX Operating System Technical Reference*.

---

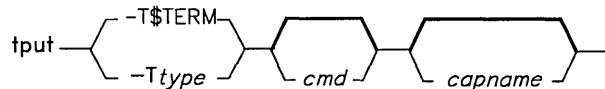
# tput

---

## Purpose

Queries the **terminfo** file.

## Syntax



OL805398

## Description

The **tput.** command uses the **terminfo** file to make terminal-dependent information available to the shell. The output of **tput.** is a string if the attribute *capname* (for capability name) is of type string or an integer if the attribute is of type integer. If the attribute is of type Boolean, **tput.** simply sets the exit value (0 for TRUE, 1 for FALSE), and produces no other output.

## Flags

- Ttype** Indicates the type of work station. Normally, the value of *type* is supplied by the environment variable **\$TERM**.
- capname** Indicates the attribute from the **terminfo** file. For more information, see the **terminfo** file in *AIX Operating System Technical Reference*.

## Examples

1. To echo the clear-screen sequence for the current work station:  
tput clear
2. To display the number of columns for the current work station:  
tput cols
3. To display the number of columns for the 450 work station:  
tput -T450 cols



## tput

---

4. To set the shell variable `bold` to the highlight mode sequence for the current work station:

```
bold=`tput smso`
```

This might be followed by a prompt:

```
echo "${bold}Please type in your name: \c"
```

5. To set the exit value to indicate if the current work station is a hardcopy terminal:

```
tput hc
```

## Files

|                                    |                            |
|------------------------------------|----------------------------|
| <code>/usr/lib/terminfo/?/*</code> | Terminal descriptor files. |
| <code>/usr/include/term.h</code>   | Definition files.          |
| <code>/usr/include/curses.h</code> |                            |

## Related Information

The following command: “`stty`” on page 717.

The `terminfo` file in *AIX Operating System Technical Reference*.

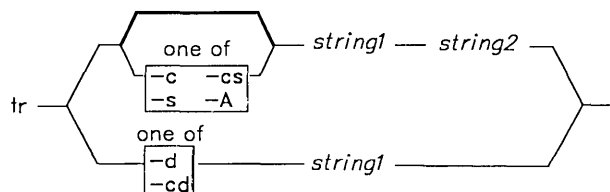
---

**tr**


---

**Purpose**

Translates characters.

**Syntax**

OL805278

**Description**

The **tr** command copies characters from the standard input to the standard output with substitution or deletion of selected characters. Input characters from *string1* are replaced with the corresponding characters in *string2*. **tr** cannot handle an ASCII NUL (000) in *string1* or *string2*; it always deletes NUL from the input.

Abbreviations that can be used to introduce ranges of characters or repeated characters are:

- [a-z]**        Stands for a string of characters whose ASCII codes run from character **a** to character **z**, inclusive.
- [a\*num]**     Stands for *num* repetitions of **a**. *num* is considered to be in decimal unless the first digit of *num* is **0**; then it is considered to be in octal.

Use the escape character **\** (backslash) to remove special meaning from any character in a string. Use the **\** followed by 1, 2, or 3 octal digits for the ASCII code of a character.

**Flags**

- A**    Translates on a byte-by-byte basis. When you specify this flag, **tr** does not support extended characters.
- c**    Complements (inverts) the set of characters in *string1* with respect to the universe of characters whose ASCII codes are 001 through 377 octal, if you specify **-A**, and all characters, if you do not specify **-A**.

- d Deletes all input characters in *string1*.
- s Changes characters that are repeated output characters in *string2* into single characters.

## Examples

1. To translate braces into parentheses:  

```
tr '{}()' <textfile >newfile
```

This translates each { to ( and each } to ). All other characters remain unchanged.
2. To translate lowercase characters to uppercase:  

```
tr '[a-z]' '[A-Z]' <textfile >newfile
```
3. This is what happens if the strings are not the same length:  

```
tr '[0-9]' '#' <textfile >newfile
```

This translates each 0 to a # (number sign).

**Note:** If the two character strings are not the same length, then the extra characters in the longer one are ignored.
4. To translate each digit to a #:  

```
tr '[0-9]' '[#*]' <textfile >newfile
```

The \* tells tr to repeat the # enough times to make the second string as long as the first one.
5. To translate each string of digits to a single *num*:  

```
tr -s '[0-9]' '[#*]' <textfile >newfile
```
6. To translate all ASCII characters that are *not* specified:  

```
tr -c '[-~]' '[A-~]?' <textfile >newfile
```

This translates each non-printing ASCII character to the corresponding control key letter (\001 translates to A, \002 to B, etc.). ASCII DEL (\177), the character that follows ~ (tilde), translates to ?.
7. To create a list of the words in a file:  

```
tr -cs '[a-z][A-Z]' '[\012*]' <textfile >newfile
```

This translates each string of nonalphanumeric characters to a single new-line character. The result is a list of all the words in *textfile*, one word per line.

## Related Information

The following commands: “**ed**” on page 280 and “**sh**” on page 637.

The **ascii** file in *AIX Operating System Technical Reference*.

The “Overview of International Character Support” in *Managing the AIX Operating System*.

# trace

---

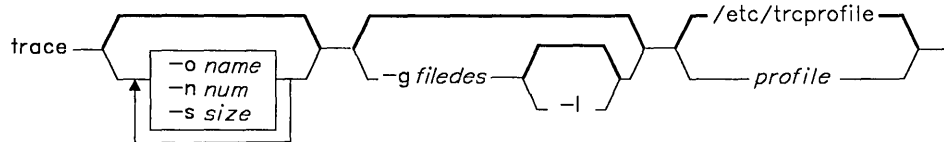
## trace

---

### Purpose

Starts the trace function.

### Syntax



OL805279

### Description

The **trace** command starts the trace function in the background. This trace function provides a base for debugging the system. **trace** monitors the occurrence of selected events in the system and records on disk important data specific to each of these events. You can format this output with the **trcrpt** command.

Any user or program that needs the trace process enabled for debugging or error determination can start **trace**. When starting **trace**, you must provide a *profile*. This allows you to tailor the output of the trace session to individual needs. The default *profile* is **/etc/trcprofile**.

There may be more than one trace profile in the file system at a time. The trace profile contains the classes of events that you can select to trace, listed by event class and by a descriptive label. See “Example” on page 770 for a sample profile. You may keep different profiles to trace different combinations of event classes. **trace** also takes additional information about the trace session from the configuration file **/etc/rasconf** (see *AIX Operating System Technical Reference* for a discussion of this file). You set the name and size of the output file in this configuration file.

In a multi-user environment, **trace** records all system events, not just events at one virtual terminal.

---

## Flags

**-g** *filedes* Indicates that this is a **generic** trace session. Generic tracing applies only to the VRM. In this type of session, events to be recorded do not necessarily have a fixed event class, but are allocated to a temporary event channel by the trace device driver, `/dev/vrmtrace`. Thus, starting a generic trace does not require a trace profile. Generic traces are started and stopped by other processes, such as communications session managers. Therefore, the interface to the demon is somewhat different. The **-g** flag is useful only when **trace** is started by another process.

The *filedes* parameter is a file descriptor from the parent process. **trace** writes this following information to this file descriptor:

- The process ID of the **trace** demon
- The address of the trace buffer
- The size (in bytes) of the trace buffer
- The temporary channel bit allocated to this event.

When tracing a generic event, the **trace** demon does not record its process ID so that it can be stopped by the **trcstop** command. Thus, more than one **trace** demon may be running at any time, but there may be as many as seven traces in the system at once (one normal trace and from one to six generic traces).

Use the **trc\_start** and **trc\_stop** subroutines to start and stop a generic trace.

**-l** Indicates that the VRM trace device driver should log only the last buffer filled before the **trace** demon stops. This flag is valid only during a generic trace (**-g**).

**-o** *name* Specifies the name of the log file into which the **trace** demon stores the trace data. For generic traces (**-g**), this name must be different from the default file name specified in the configuration file `/etc/rasconf`.

**-n** *num* Specifies the number of entries in the trace buffer. **trace** multiplies this number by the size of the entries (see the **-s** flag) and uses the resulting value to size the trace buffer. If you do not specify this flag, **trace** uses the buffer size specified in the configuration file `/etc/rasconf`.

**-s** *size* Specifies the size (in bytes) of the entries that the **trace** demon will be handling. The default size is 40 bytes. The size can be no less than 20, which is the number of bytes in the trace header for each entry. All entries must be the same size in a particular trace log file.

## trace

---

### Example

```
*****
* SYSTEM TRACE PROFILE
*****
* To set trace on for an event class, remove the comment mark (*) from the
* first column of the line containing the event you wish to trace.
* Add a comment mark (*) in the first column of lines containing event types
* you wish to stop tracing.

***** Event
*      Type      Description

*****
          Applications

*****
          AIX Extensions
*      36      Config

*****
          AIX System Calls
*      60      Shared Memory
*      61      Messages
*      62      Semaphores
*      63      Signals
*      64      Time
*      65      File System
*      66      File Handling
*      67      Directory Handling
*      68      Process

*****
          VRM Components
*      100     SVC Handler
*      110     Async/5080 Peripherals
*      112     Async/5080 Peripheral Interrupts
*      113     Virtual Terminal Manager
*      114     Keyboard Interrupts
*      115     Locator Interrupts

*      150     User-Defined Events
```

## Files

|                                   |                                                    |
|-----------------------------------|----------------------------------------------------|
| <code>/etc/trcprofile</code>      | Default profile.                                   |
| <code>/usr/adm/ras/trcfile</code> | Output file defined in <code>/etc/rasconf</code> . |
| <code>/etc/rasconf</code>         | Configuration file.                                |

## Related Information

The following commands: “**trcstop**” on page 774 and “**trcrpt**” on page 772.

The **rasconf** configuration file in *AIX Operating System Technical Reference*.

The discussion of **trace** in *AIX Operating System Programming Tools and Interfaces*.



# trcrpt

---

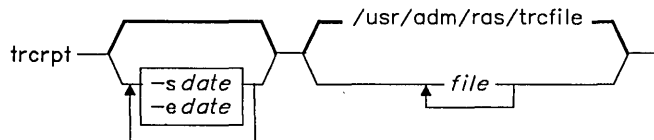
## trcrpt

---

### Purpose

Formats a report from the trace log file.

### Syntax



OL805280

### Description

The **trcrpt** command writes to standard output a chronological listing in readable format of the trace log *file* or *files* specified. You can specify a maximum of 10 log files. If you do not specify any files, **trcrpt** reads */etc/rasconf* for a file name. This name is usually */usr/adm/ras/trcfile*.

### Flags

- e *date* Ends the report time with entries on or before *date*. The format of *date* is the same as the **date** command, *MMddhhmmyy*.
- s *date* Starts the report with entries on or later than *date*. The format of *date* is the same as the **date** command, *MMddhhmmyy*. If you do not specify this flag, **trcrpt** formats the entire log file.

### Example

To format a trace log file:

```
trcrpt -s0109100384 -e0109100584 /u/dave/trc_log | print
```

This formats the log file */u/dave/trc\_log*, starting with entries from January 09, 1984 at 10:03 and ending at 10:05. It pipes the formatted output to the print queue.

## Files

|                         |                          |
|-------------------------|--------------------------|
| /usr/adm/ras/trcfile    | Default log file.        |
| /etc/trcfmt             | Trace format file.       |
| /usr/adm/ras/.trcevents | Trace event types table. |

## Related Information

The following commands: “**trace**” on page 768 and “**trcstop**” on page 774.

The **rasconf** file in *AIX Operating System Technical Reference*.

The discussion of **trcrpt** *AIX Operating System Programming Tools and Interfaces*.

# trcstop

---

## trcstop

---

### Purpose

Stops the trace function.

### Syntax

trcstop —

OL805223

### Description

The **trcstop** command sends a Software Terminate signal to the **trace** background process. This gracefully ends **trace** and forces cleanup.

### Files

/tmp/trc\_PIDs

### Related Information

The following commands: “**trace**” on page 768 and “**trcrpt**” on page 772.

The discussion of **trcstop** in *AIX Operating System Programming Tools and Interfaces*.

---

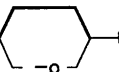
## trcupdate

---

### Purpose

Updates trace format templates.

### Syntax

```
trcupdate — file — 
```

OL805399

### Description

The **trcupdate** command adds, replaces, or deletes trace report format templates in the files **/etc/trcfmt** and **/usr/adm/ras/.trcevents** and the event types in file **/etc/trcprofile**. **trcupdate** creates three undo files in the current directory named **file.undo.trc**, **.trcevents.undo.evt**, and **file.undo.pro**. These undo files can be used as input to **trcupdate** with the **-o** (override) flag to undo the changes **trcupdate** has just made. The **trcupdate** command reads three files named **file.trc**, **file.evt**, and **file.pro**. The **trc** file contains trace format templates; the **evt** file contains trace event types and their corresponding hook IDs; the **pro** file contains the event type line for the trace profile.

The first field of each template contains an operator:

- + To add or replace a template
- To delete a template.

If the operation is **+**, then the following fields contain the template to be replaced. The hook ID of the template is also added to the **/usr/adm/ras/.trcevents** file, and the event type line is added to the trace profile **/etc/trcprofile**. If the operation is **-**, then the second field contains the hook ID of the template to delete. That hook ID is also deleted in **/usr/adm/ras/.trcevents**, and the event type line is deleted from **/etc/trcprofile**. When adding or replacing, **trcupdate** compares the version numbers of each input template with the version number of the existing template of the same hook IDs. If the version number of the input template is later, it replaces the old template with the input template. If the template does not already exist, then it is added to the file. The input file **must** contain an identifier line on the first line: **\* /etc/trcfmt** or **trcupdate** rejects the input file.

The **file.evt** file contains a table of trace system event types and hook IDs that fall under these types. **trcupdate** reads in the file **/usr/adm/ras/.trcevents** and adds in any hook IDs from **file.evt** that are not already accounted for or reassigns/deletes hook IDs to the

## trcupdate

---

event type given in the update file. The first line of the event/hook update file *must* be:  
\* /ras/.trcevents or **trcupdate** rejects the input file.

The *file.pro* contains the lines that are to be added to or deleted from */etc/trcprofile*. **trcupdate** reads */etc/trcprofile* and adds or deletes the specified event type line from *file.pro*. The first line of the event type file *must* be: \* /etc/trcprofile or **trcupdate** rejects the input file.

## Flags

-o Does no version number checking.

## Examples

1. The following is a sample **trc** file:

```
* /etc/trcfmt
+ 355 1.0 new_fmt
- 351
- 352
- 353
```

2. The following is a sample **evt** file:

```
* ras/.trcevents
350 355 356 357
```

## Files

```
/etc/trcfmt
/usr/adm/ras/.trcevents
file.evt
file.undo.evt
file.trc
file.undo.trc
file.pro
file.undo.pro
```

## Related Information

The following command: “**trcrpt**” on page 772.

*AIX Operating System Programming Tools and Interfaces.*

## true

---

### Purpose

Returns an exit value of zero.

### Syntax

```
true —|  
false —|
```

OL805064

### Description

The **true** command returns a zero exit value. The **false** command returns a nonzero value. These commands are usually used in input to the **sh** command.

### Example

To construct an infinite loop in a shell procedure:

```
while true  
do  
    date  
    sleep 60  
done
```

This shell procedure displays the date and time once a minute. To stop it, press **INTERRUPT (Alt-Pause)**.

### Related Information

The following command: “**sh**” on page 637.

## tsort

---

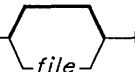
## tsort

---

### Purpose

Sorts an unordered list of ordered pairs (a topological sort).

### Syntax

tsort  *file*

OL805224

### Description

The **tsort** command reads from *file* or standard input an unordered list of ordered pairs, it builds a completely ordered list, and writes it to standard output.

The input *file* should contain pairs of nonempty strings separated by blanks. Pairs of different items indicate a relative order. Pairs of identical items indicate presence, but no relative order. You can use **tsort** to sort the output of the **lorder** command.

If *file* contains an odd number of fields, **tsort** writes the error message `Odd data`

### Example

To create a subroutine library:

```
lorder charin.o scanfld.o scan.o scanln.o \  
| tsort | xargs ar qv libsubs.a
```

This creates a subroutine library named `libsubs.a` that contains `charin.o`, `scanfld.o`, `scan.o`, and `scanln.o`. The ordering of the object modules in the library is important. The **ld** command requires each module to precede all the other modules that it calls or references. The **lorder** and **tsort** commands together add the subroutines to the library in the proper order.

---

Suppose that `scan.o` calls `scanfld.o` and `scanln.o`. `scanfld.o` also calls `charin.o`. First, the **lorder** command creates a list of pairs that shows these dependencies:

```
charin.o charin.o
scanfld.o scanfld.o
scan.o scan.o
scanln.o scanln.o
scanfld.o charin.o
scanln.o charin.o
scan.o scanfld.o
```

Next, the `|` (vertical bar) sends this list to the **tsort** command, which converts it into the ordering we need:

```
scan.o
scanfld.o
scanln.o
charin.o
```

Note that each module precedes the module it calls. `charin.o`, which does not call another module, is last.

The second `|` then sends this list to **xargs**, which constructs and runs the following **ar** command:

```
ar qv libsubs.a scan.o scanfld.o scanln.o charin.o
```

This **ar** command creates the properly ordered library.

## Related Information

The following commands: “**ar**” on page 58, “**lorder**” on page 457, and “**xargs**” on page 857.



**ttt**

---

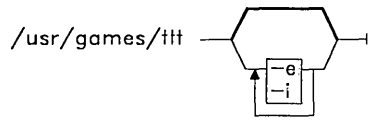
**ttt**

---

## Purpose

Plays tic-tac-toe.

## Syntax



OL805282

## Description

The `ttt` game plays the popular X and O game. This is a learning version, but it learns slowly. It loses nearly 80 games before completely mastering the game.

## Flags

- `-e` Increases the speed of the learning.
- `-i` Displays the instructions prior to the start of the game.

To quit the game, press INTERRUPT (**Alt-Pause**) or END OF FILE (**Ctrl-D**).

## Files

`/usr/games/ttt.a` Learning file.

---

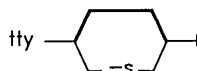
## tty

---

### Purpose

Writes to standard output the full path name of your work station.

### Syntax



OL805283

### Description

The `tty` command writes the name of your work station to standard output.

### Flag

- `-s` Suppresses reporting the path name. The exit value has the following possible meanings:
  - 0** Standard input is a work station.
  - 1** Standard input is not a work station.
  - 2** Invalid flags specified.

If your standard input is not a work station and you do not specify the `-s` flag, you get the message `not a tty`.

### Examples

1. To display full path name of your work station:

```
tty
```
2. To test whether or not the standard input is a work station:

```
if tty -s
then
    echo 'Enter the text to print:' >/dev/tty
fi
print
```

## **tty**

---

If the standard input is a work station, this displays the message Enter the text to print: as a prompt and prints the text that the user types.

If the standard input is not a work station, this displays nothing. It merely prints the text read from the standard input.

The echo . . . >/dev/tty displays the prompt on the screen even if you redirect the standard output of the shell procedure. This way the prompt is never written into an output file. The special file **/dev/tty** always refers your work station, although it also has another name like **/dev/console** or **/dev/tty2**.

## turnon

---

### Purpose

Turns on execute permission for games.

### Syntax

turnon —l

OL805405

turnoff —l

OL805406

### Description

The **turnon** and **turnoff** commands are shell procedures that set the permission codes of files in the **/usr/games** directory. You must be operating with superuser authority to run this command.

The **turnon** command looks for files with permissions set to 000 and sets them to 111 (execute permission for all users).

The **turnoff** command looks for files in **/usr/games** whose permissions are set to 111 and sets these permissions to 000.

If you install any new games in the **/usr/games** directory, set their permissions to 111.

## **ugtable**

---

## **ugtable**

---

### **Purpose**

Accesses the Distributed Services Network User/Groups Table.

### **Syntax**

ugtable —

OL805469

### **Description**

The **ugtable** command lets you build, examine, or change the Distributed Services Network User/Group Table. Only members of the system group or users operating with superuser authority can use **ugtable** to change the state of the Distributed Services Network User/Group Tables (see “**su**” on page 724). Other users can use **ugtable** to browse the Network User/Groups Table.

### **Related Information**

“Getting Started With Distributed Services Configuration Menus” in *Managing the AIX Operating System*.

---

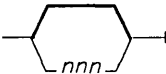
# umask

---

## Purpose

Sets file-creation permission code mask.

## Syntax

umask 

OL805286

## Description

The **umask** command sets your file-creation mask to *nnn*, three octal digits that represent the read/write/execute permissions for owner, group, and others, respectively. When you create a file, the system ANDs the complement of *nnn* to the file-creation permission code, in effect removing the corresponding permissions. (See “**chmod**” on page 128 for more information on file and directory permission codes.)

If you do not specify *nnn*, **umask** displays the current value of your file-creation permission code mask. The initial system mask (set in */etc/profile*) is 022.

## Examples

1. To display the current file creation mask:

```
umask
```

2. To prevent other people from writing into your files:

```
umask 022
```

This sets the file creation mask to 022, which takes away write permission for group members and others. Now a file that would normally be created with permission code 777 has code 755, and a file normally 666 has 644.

3. To prevent other people from using your files:

```
umask 077
```

This sets the file creation mask to 077, which removes read, write, and execute permission for group members and others. Now a file that would normally be created with permission code 666 has code 600.

## umask

---

### Related Information

The following commands: “**chmod**” on page 128 and “**sh**” on page 637.

The **creat**, **chmod**, **mknod**, **open**, and **umask** calls in *AIX Operating System Technical Reference*.

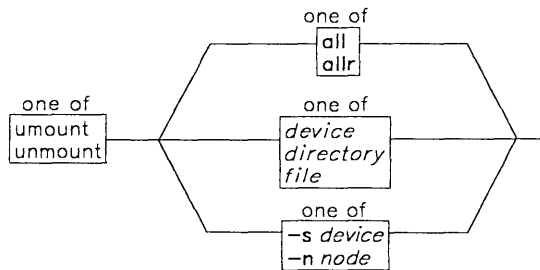
The discussion of tailoring the user environment in *Managing the AIX Operating System*.





**umount****umount****Purpose**

Makes a file system unavailable for use.

**Syntax**

OL805225

**Description**

The **umount** command unmounts a previously mounted file system, directory, or file. Processing on the file system, directory, or file completes and it is unmounted. Members of the system group and users operating with superuser authority can issue any **umount** command. Other users can unmount any directory or file that they have mounted. For local mounts, you can specify the file system, directory, or file as either the *directory* or *device* on which it is mounted. If you specify **all**, **umount** unmounts all mounted file systems. For remote mounts, specify the directory of the file as *directory*. If you specify **allr**, **umount** unmounts all remote mounts.

**Flags**

- n node** Specifies the remote node for the unmount. *node* can be either a nickname or a node ID. The **umount -n node** command unmounts all remote mounts made from *node*.
- s** Prohibits the use of the **/etc/mnttab** file if it is damaged or not writable. If you use this flag, you must specify the name of the *device* to be unmounted.

**Note:** You cannot use the **umount** command on a device that is in use. A device is in use if any file is open for any reason or if a user's current directory is on that device.

## Examples

1. To unmount a diskette drive:  
umount /dev/fd0
2. To unmount the device mounted on **/diskette0**:  
umount /diskette0
3. To unmount all mounts from a remote node:  
umount -n nodeA

## Files

/etc/mnttab      Table of currently mounted file systems.

## Related Information

The following command: “**mount**” on page 498.

The **mount**, **umount**, **vmount**, **uvmount**, and **mntctl** system calls and the **mnttab** file in *AIX Operating System Technical Reference*.

# uname

---

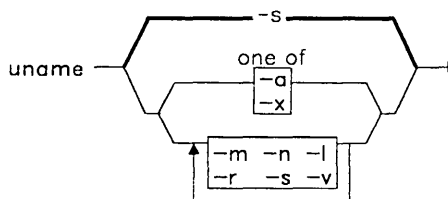
## uname

---

### Purpose

Displays the name of the current operating system.

### Syntax



OL805287

### Description

The **uname** command writes to the standard output the name of the operating system that you are using.

### Flags

- a Displays all information specified with the **-m**, **-n**, **-r**, **-s**, and **-v** flags.
- l Displays the LAN network number.
- m Displays the type of hardware running the system.
- n Displays the name of the node (this may be a name that the system is known by to a communications network).
- r Displays the release number of the operating system.
- s Displays the system name. (This flag is on by default.)
- v Displays the operating system version.
- x Displays the information specified with the **-a** flag and the LAN network number.

If you enter a flag that is not valid, **uname** exits with an error message, an error return status, and no output.

## **Example**

To display the complete system name and version banner:

```
uname -a
```

## **Related Information**

The **uname** system call in *AIX Operating System Technical Reference*.

# unget

---

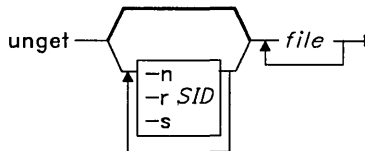
## unget

---

### Purpose

Cancels a previous **get** command.

### Syntax



OL805284

### Description

The **unget** command allows you to restore a g-file created with a **get -e** before the new delta is created, and therefore discarding the changes (see “**get**” on page 359 and “**delta**” on page 236). If you specify a - (hyphen) in place of *file*, standard input is read, and each line of standard input is interpreted as the name of an SCCS file. **unget** continues to take input until it reaches an end of file character, which is a **Ctrl-D** if input is from the keyboard.

If you specify a directory in place of *file*, **unget** performs the requested actions on all SCCS files (those files with the **s.** prefix).

### Flags

Each flag or group of flags applies independently to each named file.

- n** Prevents the automatic deletion of the g-file. This flag allows you to retain the edited version of the file without making a delta.
- rSID** Specifies the new delta that would have been created by the next use of the **delta** command. You must use this flag if you have two or more pending deltas to the file under the same login name. You can look at the p-file to see if you have more than one delta pending to a particular SID under the same login name. The **SID** specification must unambiguously specify only one SID to discard, or **unget** displays an error message and stops running.
- s** Suppresses writing the deleted SID to standard output.

## Example

To discard the changes you have made to an SCCS file after doing a **get -e**:

```
unget s.prog.c
```

## Related Information

The following commands: “**delta**” on page 236, “**get**” on page 359, and “**sact**” on page 609.

The **sccsfile** file in *AIX Operating System Technical Reference*.

The discussion of SCCS in *AIX Operating System Programming Tools and Interfaces*.

# uniq

---

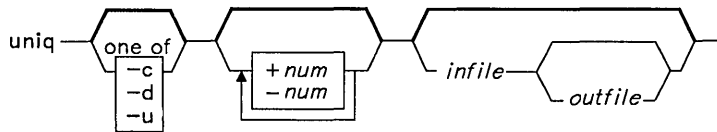
## uniq

---

### Purpose

Deletes repeated lines in a file.

### Syntax



OL805285

### Description

The **uniq** command reads standard input or *infile*, compares adjacent lines, removes the second and succeeding occurrences of a line, and writes to standard output or the specified file *outfile*. *infile* and *outfile* should always be different files. Repeated lines must be on consecutive lines in order to be found. You can arrange them with the **sort** command (see page 672) before processing.

### Flags

- c** Precedes each output line with a count of the number of times each line appears in the file. This flag supersedes **-d** and **-u**.
- d** Displays only the repeated lines.
- u** Displays only the unrepeated lines.
- num** Skips over the first *num* fields. A field is a string of nonspace, nontab characters separated by tabs and or spaces from adjacent data on the same line.
- + num** Skips over the first *num* characters. Fields specified by *num* are skipped before characters.

### Related Information

The following commands: “**comm**” on page 144 and “**sort**” on page 672.

---

# units

---

## Purpose

Converts units in one measure to equivalent units in another measure.

## Syntax

units —|

OL805226

## Description

The **units** command converts quantities expressed in one measurement to their equivalents in another. **units** is an interactive command. It prompts you for the unit you want to convert from and the unit you want to convert to (see “Examples” on page 794). This command only does multiplicative scale changes. That is, it can convert from one value to another only when the conversion is done with a multiplication factor. For example, it can not convert between degrees Fahrenheit and degrees Celsius, because 32 must be added or subtracted in the conversion.

You can specify a quantity as a multiplicative combination of units, optionally preceded by a numeric multiplier.

Indicate powers by suffixed positive integers and division by / (slash).

The **units** command recognizes **lb** as a unit of mass, but considers **pound** to be the British pound sterling. Compound names are run together (such as **lightyear**). Prefix British units differing from their American counterparts with **br** (**brgallon** for instance). The file **/usr/lib/unittab** contains a complete list of the units that the **units** command uses.

Most familiar units, abbreviations, and metric prefixes are recognized, together with the following:

|              |                                        |
|--------------|----------------------------------------|
| <b>pi</b>    | Ratio of circumference to diameter     |
| <b>c</b>     | Speed of light                         |
| <b>e</b>     | Charge on an electron                  |
| <b>g</b>     | Acceleration of gravity                |
| <b>force</b> | Same as <b>g</b>                       |
| <b>mole</b>  | Avogadro's number                      |
| <b>water</b> | Pressure head per unit height of water |
| <b>au</b>    | Astronomical unit.                     |



## units

---

### Examples

To start the **units** command, enter:

```
units
```

Now you can try the following examples. In these examples, the text that you enter is shown in **bold type** and the output from **units** is shown in non-bold type.

1. To display conversion factors:

```
you have: in  
you want: cm  
          * 2.540000e+00  
          / 3.937008e-01
```

The output from **units** tells you to multiply the number of inches by 2.540000e+00 to get centimeters, and to multiply the number of centimeters by 3.937008e-01 to get inches.

These numbers are in standard exponential notation, so 3.937008e-01 means  $3.937008 \times 10^{-1}$ , which is the same as 0.3937008. The second number is always the reciprocal of the first. That is,  $2.54 = 1 \div 0.3937008$ .

2. To convert a measurement to different units:

```
you have: 5 years  
you want: microsec  
          * 1.577846e+14  
          / 6.337753e-15
```

The output shows that **5 years** equals  $1.577846 \times 10^{14}$  microseconds, and that one microsecond equals  $6.337753 \times 10^{-15}$  years.

3. To give fractions in measurements:

```
you have: 1|3 mi  
you want: km  
          * 5.364480e-01  
          / 1.864114e+00
```

The | (vertical bar) indicates division, so **1|3** means one-third. This shows that one-third mile is the same as 0.536448 kilometers.

4. To include exponents in measurements:

```
you have: 1.2-5 gal  
you want: floz  
          * 1.536000e-03  
          / 6.510417e+02
```

The expression **1.2-5 gal** stands for  $1.2 \times 10^{-5}$ . Do *not* type an **e** before the exponent. This example shows that  $1.2 \times 10^{-5}$  (0.000012) gallons equal  $1.536 \times 10^{-3}$  (0.001536) fluid ounces.

5. To specify complex units:

```
you have: gram centimeter/second2
you want: kg-m/sec2
          * 1.000000e-05
          / 1.000000e+05
```

The units **gram centimeter/second2** mean “grams × centimeters ÷ second<sup>2</sup>.” Similarly, **kg-m/sec2** means “kilograms × meters ÷ sec<sup>2</sup>,” which is often read as “kilogram-meters per seconds squared.” Note that you can show multiplication of units with a - (hyphen) or with a blank.

6. If the units you specify after “you have” and “you want” are incompatible:

```
you have: ft
you want: lb
conformability
          3.048000e-01 m
          4.535924e-01 kg
```

The message `conformability` means that the units you specified cannot be converted. Feet measure length, and pounds measure mass, so converting from one to the other doesn’t make sense. Therefore, the **units** command displays the equivalent of each value in standard units.

In other words, this example shows that one foot equals 0.3048 meters and that one pound equals 0.4535924 kilograms. **units** shows the equivalents in meters and kilograms because the command considers these units to be “standard” measures of length and mass.

## Files

`/usr/lib/unittab`

# updatep

---

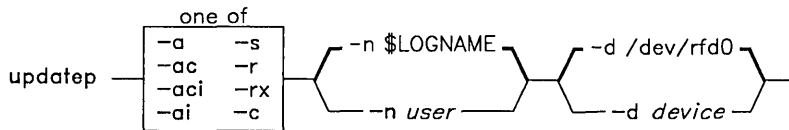
## updatep

---

### Purpose

Updates one or more programs.

### Syntax



OL805392

### Description

**Warning:** Before you apply or reject an update, you must restart your system and be sure that you are not running any programs and that no other work stations are enabled.

The **updatep** command controls the update process for one or more programs. It also lets you determine the status of pending program updates and provides documentation about the updates. You must be a member of the system group to run this command.

The **updatep** command supports an apply/commit/reject philosophy. To apply one or more programs, you use the **-a** or the **-ai** flags. You must then use either the **-c** flag to commit the program or the **-r** flag to reject the program. Normally you should not use **-r** until you have tested the program on your system. If you specify **-ac** or **-aci**, you can apply and commit in one operation. The **-r** flag must be used separately. During an apply, **updatep** normally saves the current versions of files that are being updated. If needed, these files can be used to do a recovery or reject.

You are responsible for reserving update save space in the **/usr** file system. **updatep** checks to insure that there is adequate save space in **/usr** before it applies an update. If there is insufficient free space, **updatep** gives you the option of either ending the command or of allowing it to continue. If you end the command, you can take action to increase the free space in your **/usr** file system. If you continue, no current versions of files will be saved, and **updatep** will automatically commit the update, even though you may not have requested a commit originally. Normally, you should reserve 4000 blocks (2 megabytes) of free space in the **/usr** file system for updates.

You cannot use INTERRUPT (Alt-Pause) to stop the **updatep** command. To stop **updatep**, press QUIT WITH DUMP (Ctrl-V). This should be used only in extreme circumstances since the state of the system cannot be predicted. For example:

- The write-verify feature may be left on for all minidisks. See “**verify**” on page 830
- All terminals other than the console may be disabled. See “**penable**” on page 550.
- Some update control files may need to be deleted.

## Flags

- a[i]** Applies the updates for one or more programs. If there is a pending update for any program on the system, **updatep** does not permit an apply. You must either commit or reject all pending updates before it accepts another update apply.
- The **updatep** command asks you to select the program you wish to update. After you select a program, **updatep** runs the **inudocm** command for any specific update instructions. If it finds any, it copies them into the file */usr/lpp/pgm-name/ui.vv.rr.lll.*, where *vv* is the version, *rr* the release, and *lll* the level of the program. Normally you should review instructions before continuing. To restart the update procedure and ignore the check for existing update instructions, enter **updatep -ai** or **updatep -aci**.
- The **updatep** command applies the update for each program by running **inuupdt** for each name. After each update, it deletes the */usr/lpp/pgm-name/inst\_updt* directory. It then runs **inudocm** to check for any update documentation. If there is information for a manual, **updatep** copies it into the file */usr/lpp/pgm-name/me.vv.rr.lll* and writes a message.
- c** Accepts a previous update apply. **updatep** presents selection information for programs that have pending updates. You select the programs that you want to commit.
- Any programs that you apply as a group must be committed as a group. Management control information about the update changes to indicate that the program is accepted. **updatep** deletes the directory that contains the update recovery information, */usr/pgm-name/inst\_updt.save*.
- d device** Specifies the input device name. The default input device is */dev/rfd0*.
- n user** Lets you specify a name in the program history file that is responsible for the program. The default is the value of the system variable **\$LOGNAME**. If you specify *user*, the first 8 nonblank characters are stored in the program history file.
- r** Rejects a previous update apply for one or more programs. **updatep** presents selection information for the programs that have pending updates. You select the programs to reject.

## updatep

---

programs that are grouped together by the system must be rejected or applied as a group. Specify **-r** without **-x** if you want automatic recovery of saved files. If you *do* specify the **-x** flag, the management control information about the update reflects that the update is rejected, but **updatep** does not recover saved files. You should look at the information in `/usr/lpp/pgm-name/inst_updt/save` to recover the necessary files. This flag should be used only by someone very knowledgeable about the system.

- s** Writes status information about all pending program updates.
- x** Cancels the automatic recovery of saved files (use with **-r**).

The **updatep** command receives the following exit codes indirectly from **update** through **inuupdt**:

- 0** Normal return, no errors indicated.
- 2** Use the **sync** command to update the super blocks, i-nodes, and delayed block I/O and then restart the system.
- 3** Build the kernel, then update the superblocks, i-nodes, and delayed block I/O (sync) and shut down the VRM.
- 4** Use the **cfgaply** subroutine to build the kernel. Use the **sync** command to update the super blocks, i-nodes, and delayed block I/O and then restart the system.
- 5** Installation cancelled without errors.
- 6** Update superblocks, i-nodes, and delayed block I/O (sync), then shut down the VRM.
- 7** Update cancelled by update procedure, recovery needed.

If it receives any other exit code, it runs the recovery function **inurecv**. If the system cannot run **updatep**, it returns an exit code of 1.

## Internal Commands

The **updatep** command uses the **inudocm** command for update documentation control. It uses the **inuupdt** command to apply an update to a single program. **inuupdt** runs a program-provided update procedure, **update**. **updatep** passes the following parameters to the **update** procedure:

- The full path name of the apply-list.
- The full path name of the device (file) where the update information is stored in **backup** format.

In addition to the commands discussed here, program-provided update procedures can use all of the internal commands discussed under “**installp**” on page 402. Since they are internal commands, they do a minimum validation of input parameters. Their purpose is to provide common code for functions frequently needed by most program-provided procedures. Since these internal commands function as subcommands, they return exit values rather than issue error messages. However, messages may come from other system commands that they run. C Language programmers of update procedures that call these commands can use the `/usr/include/inu21.h` file to define the return codes for them.

## inudocm

The **inudocm** command is normally used as an internal command to get copies of specific update instructions or manual errata information that you can print out. There may be cases, however, when you would enter this command from the command line (for example, if you have misplaced the manual errata information that came with a previous update). You must be a member of the system group to run this command.

The **inudocm** command has the following syntax

```
inudocm -eu [-d device] [pgm-name] [level] [-f file]
```

where *pgm-name* specifies the name of the program being checked. It must be specified unless you use the **-f** flag. It can be a maximum of 8 characters. *level* specifies the current level of *pgm-name*. This value must be identical to the level value for the last committed entry in `/usr/lpp/pgm-name/lpp.hist`. It must be specified unless you use the **-f** flag.

### Flags

- d device** Restores *file* from this *device*. *device* must be the full path name of a device special file. The default *device* is `/dev/rfd0`. You must not specify this flag if you use the **-f** flag.
- e** Requests the existing manual errata information for *pgm-name* from *level*. If you select this flag, **inudocm** uses the **ar x** command to extract the archive file `/usr/sys/inst_updt/pgm-name_erata`. (If this file is not present, no information is available.) **inudocm** extracts any level-dependent manual errata information files if there are any more recent than the current level. Selected files are moved to `/usr/lpp/pgm-name/me.vv.rr.llll`.
- f file** Identifies a file that already contains the *pgm-name* and the *level*. Only **updatep** itself should use this flag.
- u** Requests the existing specific update instructions for *pgm-name* from *level*. If you select this flag, **inudocm** uses the **ar x** command to extract the archive file `/usr/sys/inst_updt/pgm-name_instr`. (If this file is not present, no information is available.) **inudocm** extracts any level-dependent specific update instruction files if there are any more recent than the current level. Selected files are moved to `/usr/lpp/pgm-name/ui.vv.rr.llll`.

The **inudocm** command returns the following exit values:

- 0** Normal return, no error occurred.
- 1** The system cannot run **inudocm**.
- 2** Specific update instruction files were requested but not found.
- 4** Manual errata information was requested, but none were found.
- 6** Specific update instructions and manual errata were both requested but not found.
- 201** An invalid flag was specified, or the first argument was not **-e**, **-eu**, or **-u**.
- 202** One or more parameters were missing.

## updatep

---

- 204 Too many parameters were entered.
- 250 The *level* parameter did not contain exactly 4 characters, or they were not numeric.
- 251 An error occurred while attempting to restore `/usr/sys/inst_updt/control`.
- 253 The directory `/usr/lpp/pgm-name` does not exist.

### inuupdt

The **inuupdt** command provides a common interface for applying an update to a single program. Normally, **updatep** runs **inuupdt**.

The **inuupdt** command has the following format:

```
inuupdt -d device current-level new-level pgm-name
```

where *pgm-name* is the name of a program and *current-level* specifies the current maintenance level. *new-level* is the level of the update to be applied. *pgm-name* can be a maximum of 8 characters and *current-level* must be identical to the level value in the `/usr/lpp/pgm-name/lpp.hist` file.

### Flag

**-d** *device* Updates the program from the specified *device*.

The **inuupdt** command passes the following exit values to the process that called it:

- 0 Normal return.
- 2-7 Normal return.
- 101-102 Places error code in the history file.
- 104-107 Places error code in the history file.
- 201-202 Places error code in the history file.
- 204-208 Places error code in the history file.

It returns the follows exit status values:

- 100 Unknown return code received by **inuupdt**. It changes any unknown return code to 100 and logs it in the history file.
- 103 The restore of the archive file that contains the update control list, `/usr/lpp/pgm-name/inst_updt/arp`, failed.
- 201 An invalid minus parameter was specified.
- 202 One or more parameters were missing.
- 203 Apply list does not exist or was not readable.
- 204 Too many parameters were entered.

## Files

|                                            |                      |
|--------------------------------------------|----------------------|
| <code>/usr/lpp/pgm-name/inst_updt</code>   | Temporary directory. |
| <code>/usr/lpp/inst_updt</code>            | Temporary directory. |
| <code>/usr/sys/inst_updt/updt_cntrl</code> | Temporary file.      |

|                                                |                                               |
|------------------------------------------------|-----------------------------------------------|
| <code>/usr/sys/inst_updt/pgm-name</code>       | Control file.                                 |
| <code>/usr/sys/inst_updt/inutemp.xx...x</code> | Temporary files.                              |
| <code>/usr/lpp/pgm-name/inst_updt.save</code>  | Directory for saved files.                    |
| <code>/usr/sys/inst_updt/control</code>        | Temporary file.                               |
| <code>/usr/sys/inst_updt/pgm-name_instr</code> | Update instruction library.                   |
| <code>/usr/lpp/pgm-name/ui.vv.rr.llll</code>   | Update instruction file.                      |
| <code>/usr/sys/inst_updt/pgm-name_erata</code> | Document change library.                      |
| <code>/usr/lpp/pgm-name/me.vv.rr.llll</code>   | Document change file.                         |
| <code>/usr/include/inu21.h</code>              | Error code definitions for internal routines. |

## Related Information

The following command: “**installp**” on page 402.

The **lpp.hist** file in *AIX Operating System Technical Reference*.

The discussion of updating programs in *AIX Operating System Programming Tools and Interfaces*.



## users

---

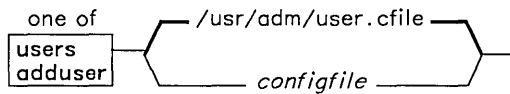
## users

---

### Purpose

Adds, deletes and changes user and group information.

### Syntax



OL805076

### Description

The **users** command lets you add, change, or delete user and group information in the **/etc/passwd** and **/etc/group** files. To use the **users** command, you must be a member of the system group or have superuser authority (see “**su**” on page 724).

**Note:** It is possible to delete a user who still owns files or to delete a group that still has members. This causes problems later, when the user name or group name is reused.

The **users** command does all of its work in temporary files. When you enter the **quit** subcommand, the temporary files become the permanent files. The old versions of **/etc/passwd** and **/etc/group** are renamed **/etc/opasswd** and **/etc/ogroup**. If **users** is ended by an INTERRUPT (**Alt-Pause**), it removes the temporary files, and the system files remain as they were before the session. Note, however, that any directories created still exist. Therefore, it may be necessary to remove these directories after sending an INTERRUPT.

For configuration, **users** uses the file **/usr/adm/user.cfile**, the file specified with *configfile*, or the default parameters that follow:

| Parameter       | Default Value  | Description                                                                                   |
|-----------------|----------------|-----------------------------------------------------------------------------------------------|
| <b>udir</b>     | /u/            | Prefix of user home directory names.                                                          |
| <b>program</b>  | null           | The name of the user login program.                                                           |
| <b>minage</b>   | null           | Minimum number of weeks that a password must be in effect before the password can be changed. |
| <b>maxage</b>   | null           | Maximum number of weeks that a password can be in effect before the password must be changed. |
| <b>siteinfo</b> | null           | Any site-specific information.                                                                |
| <b>filesize</b> | null           | Size, in blocks, of the largest file that a user can make.                                    |
| <b>gname</b>    | staff          | Name of the group that a user is initially assigned.                                          |
| <b>minid</b>    | 200            | Minimum number that can be assigned as a user or group ID.                                    |
| <b>maxid</b>    | 60000          | Maximum number that can be assigned as a user or group ID.                                    |
| <b>pfile</b>    | /etc/passwd    | Name of the password file.                                                                    |
| <b>gfile</b>    | /etc/group     | Name of the group file.                                                                       |
| <b>owner</b>    | bin            | Name of the owner of password and group files.                                                |
| <b>invalid</b>  | /usr/lib/sorry | Program for invalid accounts.                                                                 |

**Figure 7. Configuration File Parameters**

For information on how to use the **users** command, see *Managing the AIX Operating System*.

**Note:** The **/etc/passwd**, **/etc/opasswd**, **/etc/group**, **/etc/ogroup**, and **/usr/adm/user.cfile** files must all exist on the same node.

## Subcommands

- add** Adds a new user or group.
- change** Changes data for an existing user or group.
- delete** Deletes an existing user or group.
- help** Displays a summary of available commands. Entering a question mark (?) also works for help.
- invalidate** Changes a user's shell to a do-nothing program.

## users

---

**quit** Updates files and exit.  
**show** Shows information about a user or group.

The initial letter of each subcommand is recognized as the subcommand name.

## Files

|                      |                                              |
|----------------------|----------------------------------------------|
| /usr/adm/user.cfile  | Default configuration file.                  |
| /usr/adm/newuser.sys | Initialization shell file for added users.   |
| /usr/adm/newuser.usr | Initialization shell file for added users.   |
| /etc/passwd          | Password file—identifies all known users.    |
| /etc/group           | Group file—identifies all known groups.      |
| /etc/opasswd         | Saved previous version of the password file. |
| /etc/ogroup          | Saved previous version of the group file.    |

## Examples

The following is a sample `/usr/adm/user.cfile`:

```
pfile      /etc/passwd
gfile      /etc/group
owner      root
minid      200
maxid      1000
udir       /u/
program    /bin/sh
gname      staff
invalid    /usr/lib/sorry
```

## Related Information

The **group** and **passwd** files in *AIX Operating System Technical Reference*.

The discussion of **users** in *Managing the AIX Operating System*.

---

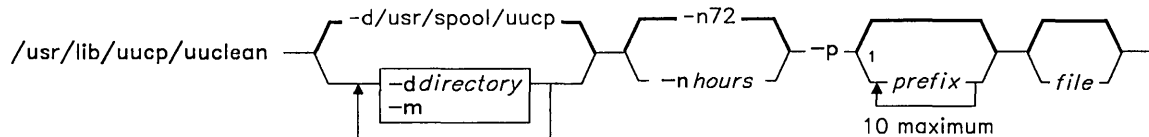
# uuclean

---

## Purpose

Deletes from the **uucp** spool directory or a named directory selected files older than a specified number of hours.

## Syntax



OL805379

## Description

The **uuclean** command scans the spool directory (or a directory named with the **-d** flag) for files with the specified prefix and deletes any of those that are older than a specified number of hours. Typically, **cron** runs **uuclean** on a regular basis.

## Flags

**-ddirectory** Cleans *directory* instead of the **uucp** spool directory (**/usr/spool/uucp**).

**-m** Notifies owners of deleted files by **mail**.

**-nhours** Deletes only files older than *hours* that match the *prefix* specified in the following **-p** flag. The default time is 72 hours.

**-pprefix** Deletes files beginning with *prefix* that are older than the number of hours selected by the **-n** flag. In a file name, a prefix is separated from the rest of the name by a . (dot.)

You can specify up to ten prefixes by repeating **-pprefix**. If you do not specify a *prefix*, **uuclean** deletes all files older than the specified time.

## Files

**/usr/lib/uucp**  
**/usr/spool/uucp**

### Related Information

The following commands: “**cron**” on page 172, “**uucp**” on page 807, and “**uux**” on page 818.

---

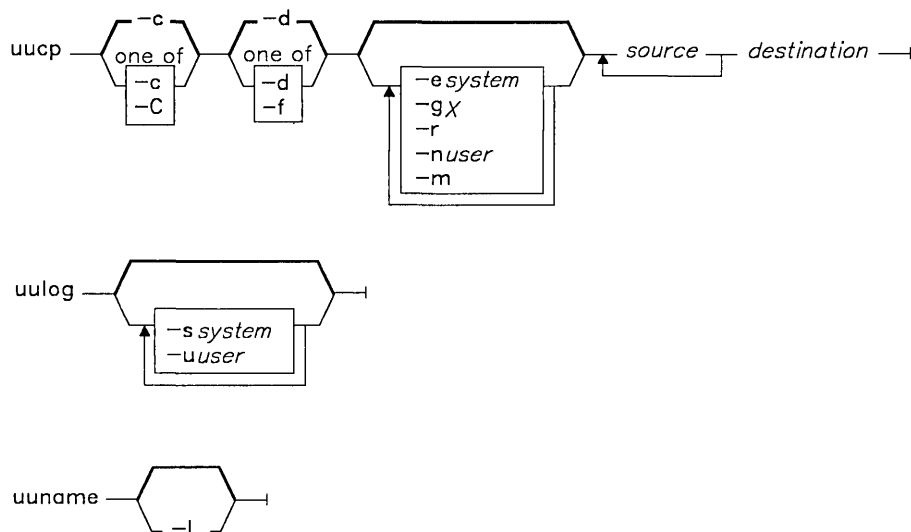
## uucp

---

### Purpose

Copies files from one AIX system to another.

### Syntax



OL805382

### Description

#### uucp

The **uucp** command copies one or more *source* files from one system to a single *destination* on another system.

A file name may be a path name on your system or may have the form:

*system!pathname*

where *system* is taken from a list of system names which **uucp** knows about. The *system* parameter can also be a list of names such as:

*system!system! . . . !system!pathname*

in which case the file is sent via the specified route (but only to a destination in **/usr/spool/uucppublic**). You must be sure that the intermediate systems in this route are willing to forward information (see *AIX Operating System Communications Guide*). In addition, if the file name on the remote system contains extended characters, all intermediate systems must support extended characters.

You can use the special shell characters `?`, `*`, and `[]` in path names. The appropriate system will expand them. However, these characters must be escaped or quoted to prevent the local shell from interpreting them.

Path names can be one of the following:

1. A full path name
2. A path name preceded by `~user`, where *user* is a login name on the specified system. **uucp** replaces this with that user's login directory.
3. A path name preceded by `~/user`, where *user* is a login name on the specified system. **uucp** replaces this with that user's directory under **/usr/spool/uucppublic**.
4. A partial path name, which is assumed to begin in the current directory.

If *destination* is a directory, **uucp** uses the last part of the *source* name.

When transmitting files, **uucp** preserves execute permissions and grants read and write permissions to everyone. (**uucp** owns the file.)

The system administrator should restrict the accessing of local files by users on other systems. Because using the `-esystem` flag to get files from other systems often fails, ask someone on the other systems to send them to you. Sending files to arbitrary *destination* path names on other systems often fails because of security restrictions.

### Flags

- c** Transfers *source* directly, rather than first copying it to the spool directory for transfer from there. (See the **-C** flag immediately following.) This flag is on by default.
- C** Copies the source file to the spool directory for transfer from there.
- d** Creates all directories as necessary. This flag is on by default.
- esystem** Sends **uucp** to the specified *system* to be executed there. This succeeds only if the other system allows **/usr/lib/uucp/uuxqt** to run **uucp**.  
**Note:** Relative file names are resolved locally before being sent.
- f** Does not create intermediate directories during file copy.

- gX** Marks the created spool file grade *X*, where *X* is an uppercase letter. **uucp** processes spooled files with alphabetically higher grades first. It makes grade **S** transfers only when the local site is the not the controlling end of the connection.
- m** Notifies the requester by **mail** when **uucp** completes the copy. The **-m** flag works for sending a number of files but for receiving only a single file.
- nuser** Notifies *user* on the other system that you sent a file.
- r** Does not start the spooling program that moves files between systems.

## **uulog**

The **uulog** command provides information about **uucp** and **uux** activities, which it reads from the file **/usr/spool/uucp/LOGFILE**. The format of the date in the log files is controlled by the environment variable **NLDATE**, if it is defined.

## **Flags**

- ssystem** Displays information about copy requests involving the named *system* (or all systems if you do not specify a *system*).
- uuser** Displays information about copy requests involving the named *user*.

## **uname**

The **uname** command writes to standard output the names of all other systems accessible to the local system.

## **Flag**

- l** Writes the local system name.

## **Files**

|                              |                                             |
|------------------------------|---------------------------------------------|
| <b>/usr/spool/uucp</b>       | Spool directory.                            |
| <b>/usr/spool/uucppublic</b> | Public directory for receiving and sending. |
| <b>/usr/lib/uucp/*</b>       | Other data and program files.               |

## **Related Information**

The following commands: “**mail**” on page 470 and “**uux**” on page 818.

The “Overview of International Character Support” in *Managing the AIX Operating System*.



# uustat

---

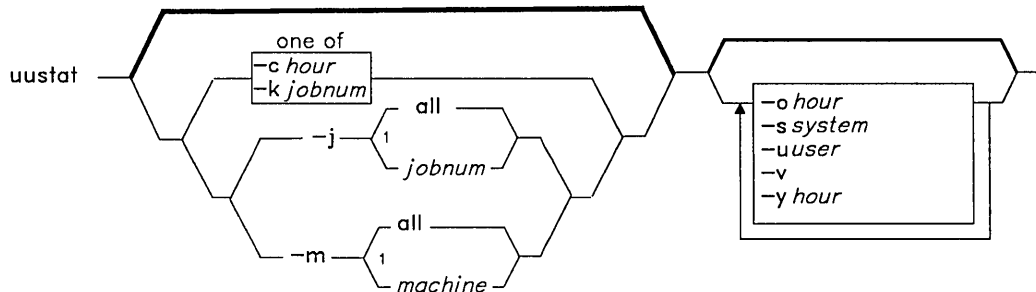
## uustat

---

### Purpose

Reports the status of and provides rudimentary job control for the **uucp** command.

### Syntax



OL805383

### Description

The **uustat** command writes to standard output the status of **uucp** commands or general status on **uucp** connections to other systems. You can also use this command with the **-k** flag to cancel **uucp** copy requests.

If you invoke **uustat** with no flags, it reports the status of all **uucp** requests issued by the current user. **uustat** reports status as:

```
job# userid system-name time status
```

where *time* is the latest status time and *status* is either an octal number or a verbose description.

### Flags

The following flags are mutually exclusive:

**-chour** Removes status entries older than *hour*. The person using this flag must either be the one who made the **uucp** request being removed or be operating with superuser authority.

- jjobnum** Reports the status of the **uucp** request *jobnum*. If you specify **all** for *jobnum*, **uustat** reports the status of all jobs.
- kjobnum** Cancels the **uucp** request *jobnum*. The person using this flag must either be the one who made the **uucp** request being canceled or be operating with superuser authority.
- mmachine** Reports the status of system *machine*. If you specify **all** for *machine*, **uustat** reports the status of all machines known to the local system.

The following flags may be combined with one of the preceding flags and with each other.

- ohour** Reports the status of all **uucp** requests older than *hour*.
- ssystem** Reports the status of all **uucp** requests that communicate with remote *system*.
- uuser** Reports the status of all **uucp** requests by *user*.
- v** Reports the **uucp** status verbosely. If you do not specify this flag, an octal status code is displayed with each **uucp** request. The octal codes are:

| OCTAL  | STATUS                                                |
|--------|-------------------------------------------------------|
| 000001 | The copy failed, but the reason cannot be determined. |
| 000002 | Permission to access local file is denied.            |
| 000004 | Permission to access remote file is denied.           |
| 000010 | Bad <b>uucp</b> command is generated.                 |
| 000020 | Remote system cannot create temporary file.           |
| 000040 | Cannot copy to remote directory.                      |
| 000100 | Cannot copy to local directory.                       |
| 000200 | Local system cannot create temporary file.            |
| 000400 | Cannot execute <b>uucp</b> .                          |
| 001000 | Copy (partially) succeeded.                           |
| 002000 | Copy finished, job deleted.                           |
| 004000 | Job is queued.                                        |
| 010000 | Job is killed (incomplete).                           |
| 020000 | Job is killed (complete).                             |

- yhour** Reports the status of all **uucp** requests younger than *hour*.

## Files

/usr/spool/uucp Spool directory  
 /usr/lib/uucp/L\_stat System status file  
 /usr/lib/uucp/R\_stat Request status file

**uustat**

---

## **Related Information**

The following command: “**uucp**” on page 807.

---

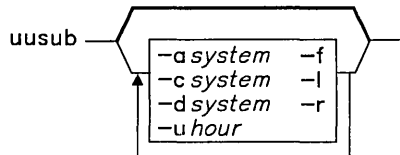
# uusub

---

## Purpose

Defines and monitors a **uucp** subnetwork structure.

## Syntax



OL805342

## Description

The **uusub** command defines a **uucp** subnetwork and monitors the connection and traffic among the systems in the subnetwork.

## Flags

- a***system* Adds *system* to the subnetwork.
- c***system* Connects to *system*. If you use **all** for *system*, **uusub.** connects to all systems in the subnetwork.
- d***system* Deletes *system* from the subnetwork.
- f** Clears the connection statistics.
- l** Reports connection statistics as follows:

```

system #call #ok time #noacu #login #nack #other
  
```

where:

|               |                                                                                                 |
|---------------|-------------------------------------------------------------------------------------------------|
| <i>system</i> | The remote system name.                                                                         |
| <i>#call</i>  | The number of times the local system tried to call <i>system</i> since the last clear was done. |
| <i>#ok</i>    | The number of successful connections.                                                           |
| <i>time</i>   | The latest successful connect time.                                                             |
| <i>#noacu</i> | The number of unsuccessful connections due to no auto call unit.                                |
| <i>#login</i> | The number of connections due to unsuccessful login.                                            |

## uusub

---

- #nack*        The number of unsuccessful connections due to no response (for example, line busy, system down).
- #other*       The number of unsuccessful connections for other reasons.
- r**        Reports the traffic volume statistics as follows.
- sysname sfile sbyte rfile rbyte*
- where:
- sysname*     The name of the system.
- sfile*        The number of files sent.
- sbyte*        The number of bytes sent during the period specified in the most recent **uusub** command with the **-uhour** flag.
- rfile*        The number of files received.
- rbyte*        The number of bytes received during the period specified in the most recent **uusub** command with the **-uhour** flag.
- uhour**     Gathers the traffic statistics over the past *hours*.

## Files

- |                               |                        |
|-------------------------------|------------------------|
| <i>/usr/spool/uucp/SYSLOG</i> | System log file.       |
| <i>/usr/lib/uucp/L_sub</i>    | Connection statistics. |
| <i>/usr/lib/uucp/R_sub</i>    | Traffic statistics.    |

## Related Information

The following commands: “**uucp**” on page 807 and “**uustat**” on page 810.

---

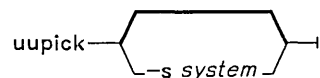
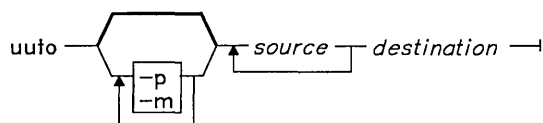
## uuto

---

### Purpose

Copies public files from one AIX system to another system with local system control of file access.

### Syntax



OL805290

### Description

#### uuto

The **uuto** command copies one or more *sources* from one system to a single *destination* on another system. **uuto** uses the **uucp** command to send files, but it allows the local system to control the file access with the **uupick** command.

#### Flags

- m** Notifies the sender by **mail** when the copy is complete.
- p** Copies the source file to the spool directory for transfer from there. The default action is to transfer a source file directly.

## uuto

---

A *source* is a path name on the local system or a path name on the system that runs the command. The *destination* has the form:

*system!user*

where *system* is the name of a system connected to the local system. (Run the **uname** command to learn which systems are connected.) *user* is the login name of someone on the specified system. The *system* parameter can also be a list of names such as:

*system!system! . . . !system!pathname*

in which case the file is sent via the specified route (but only to a destination in **/usr/spool/uucppublic**). You must be sure that the intermediate systems in this route are willing to forward information (see *AIX Operating System Communications Guide*). In addition, if the file name on the remote system contains extended characters, all intermediate systems must support extended characters.

The **uuto** command sends files to **/usr/spool/uucppublic** on *system*. Specifically, it sends files to:

*/usr/spool/uucppublic/receive/user/mysystem/files*

The **uuto** command notifies the recipient by **mail** when files arrive.

### uupick

The **uupick** command accepts or rejects the files transmitted to *user*. Specifically, **uupick** searches **/usr/spool/uucppublic** for files destined for *user*. For each entry (file or directory) found, **uupick** writes the following message to standard output.

from *system*: [file *file-name*] [dir *dirname*]?

It then waits for a response from standard input to determine the disposition of the file, as follows:

- Enter**      Goes to next entry.
- d**            Deletes the entry.
- m [dir]**     Moves the entry to directory *dir*. The default is the current directory.
- a [dir]**     Move to *dir* all the files sent from *system*.
- p**            Writes the contents of the file to standard output.
- q**            Stop.
- Ctrl-D**     Same as **q**.
- !cmd**        Runs the specified AIX command.
- \***            Writes a command summary to standard output.

### *Flags*

You can modify the action of **uupick** with the following flag:

*-ssystem* Search **/usr/spool/uucppublic** only for files sent from *system*.

### **Files**

**/usr/spool/uucppublic**          Public directory.

### **Related Information**

The following commands: “**mail**” on page 470, “**uuclean**” on page 805, “**uucp**” on page 807, “**uustat**” on page 810, and “**uux**” on page 818.



## uux

---

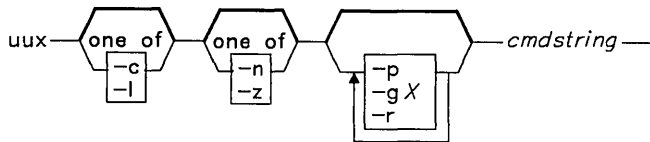
## uux

---

### Purpose

Runs a command on another AIX system.

### Syntax



OL805291

### Description

The `uux` command gathers files from various systems, runs a command on a specified system, and then writes the results to a file on a specified system. For security reasons, many installations permit `uux` to run only **mail**.

The `cmdstring` is made up of one or more arguments that look like an AIX command line, except that `cmdstring` may be prefixed by `system-name!`. The default `system-name` is the local system. The `system` parameter can also be a list of names such as:

`system!system! . . . !system!pathname`

in which case the command (or files) are searched for via the specified route. You must be sure that the intermediate systems in this route are willing to forward information (see *AIX Operating System Communications Guide*). In addition, if the file name on the remote system contains extended characters, all intermediate systems must support extended characters.

**Note:** Only the first command in a pipeline may have a `system-name!`. All other commands are executed on the system of the first command.

File names may be one of the following:

1. A full path name.
2. A path name preceded by `~user`, where `user` is a login name on the specified system. `uucp` replaces this with that user's login directory.
3. A partial path name, which is assumed to begin in the current directory.

The pattern-matching characters `?`, `*`, and `[]` in *pathname* are expanded on the appropriate system. However, the `*` may not do what you want it to do.

Quote special characters such as `<`, `>`, `,`, and `|`, either by quoting the entire *cmdstring* or by quoting the individual characters. Do not use the redirection symbols `<<` and `>>`.

The **uux** command attempts to move all files to the execution system. Use parentheses for output files so that **uux** will not try to move them.

Unless you specify the `-n` or `-z` flags, **uux** notifies you if the remote system did not allow the requested command. The response comes by remote **mail** from the other system.

## Flags

- `-c` Copies the source file directly rather than copying the file to the spool directory for transfer from there.
- `-gX` Marks the created spool file grade *X*, where *X* is an uppercase letter. **uux** processes spooled files with alphabetically higher grades first. It makes grade **S** transfers only when the local site is the noncontrolling end of the connection.
- `-l` Links source files to the spool directory rather than copying them. If the link fails, then copy them to the spool directory as usual.
- `-n` Does not notify user by **mail**, regardless of whether the command fails or completes successfully.
- `-p` Uses the standard input to the **uux** command as the standard input to *cmdstring*. `A -` (minus) has the same effect.
- `-r` Does not start the spooling program that transfers files between systems.
- `-z` Does not notify user by **mail** if the remote command completes successfully.

## Examples

1. To compare files that are located on two different systems:

```
uux "!diff usg!/usr/dan/fl pwba!/a4/dan/fl > !fl.diff"
```

This gets the `/usr/dan/fl` files located on the `usg` and `pwba` systems, runs a **diff** command, and puts the results in `fl.diff` in the local directory.

You should escape special characters such as `<`, `>`, `,`, or `|`, either by quoting the entire *cmdstring* or by quoting special characters as individual arguments.

2. To specify an output file on a different system:

```
uux a!uucp b!/u/file \{c!/u/file\}
```

## **uux**

---

This command sends the **uucp** command to system a to get /u/file from system b and send it to system c. Since **uux** attempts to get all files to the execution system, you must use parentheses to escape any output-file or nonfile arguments containing exclamation points.

### **Files**

|                      |                                                                     |
|----------------------|---------------------------------------------------------------------|
| /usr/spool/uucp      | Spool directory.                                                    |
| /usr/lib/uucp/QTcmds | List of commands permitted to be run remotely on the local machine. |
| /usr/lib/uucp/*      | Other data and programs.                                            |

### **Related Information**

The following commands: “**uuclean**” on page 805 and “**uucp**” on page 807.

---

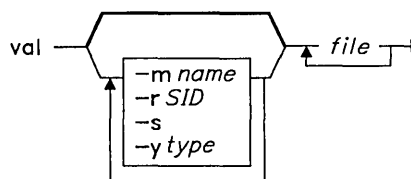
# val

---

## Purpose

Validates Source Code Control System (SCCS) files.

## Syntax



OL805292

## Description

The **val** command reads *files* and determines if the specified *file* is an **Source Code Control System (SCCS)** file meeting the characteristics specified by the flags. If you specify a - (minus) for *file*, **val** reads standard input and interprets each line of standard input as **val** flags and the name of an SCCS file. **val** continues to take input until it reaches an end-of-file character (Ctrl-D).

The **val** command displays error messages to standard output for each file processed. **val** also returns a single 8-bit code upon exit. The 8-bit code indicates possible mismatches or errors. It is interpreted as a bit string in which set bits (from left to right) are interpreted as follows:

```

bit 0 = missing file parameter
:55 1 = unknown or duplicate flag
bit 2 = damaged SCCS file
bit 3 = cannot open file or file not SCCS
bit 4 = SID is invalid
bit 5 = SID does not exist
bit 6 = %Y%, -y mismatch
bit 7 = %M%, -m mismatch
  
```

When **val** processes two or more files on a given command line or multiple command lines (when reading the standard input), a code is returned that is a logical **OR** of the codes generated for each command line and file processed. **val** can process up to 50 files on a single command line. Any number above 50 produces a dump.

## Flags

Each flag or group of flags applies independently to each named file. The flags may appear in any order

- mname* Compares the value *name* with the SCCS %M% identification keyword in *file*. See “Identification Keywords” on page 362 for more information on the %M% keyword.
- rSID* Specifies the *SID* of the *file* to be validated. The *SID* must be valid and unambiguous.
- s* Suppresses the error message normally written to standard output.
- ytype* Specifies a *type* to compare with the SCCS %Y% identification keyword in *file*. See “Identification Keywords” on page 362 for more information on the %Y% keyword.

## Related Information

The following commands: “**admin**” on page 51, “**delta**” on page 236, “**get**” on page 359, and “**prs**” on page 574.

The **scsfile** file in *AIX Operating System Technical Reference*.

The discussion of SCCS in *AIX Operating System Programming Tools and Interfaces*.

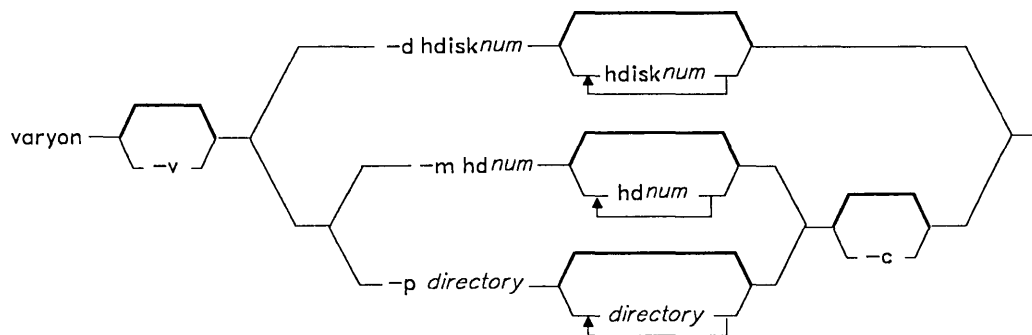
---

**varyon**

---

**Purpose**

Makes an IBM 9332 Direct Access Storage Device and the minidisks and file systems defined on it available for use.

**Syntax**

OL805455

**Description**

The **varyon** command adds an IBM 9332 Direct Access Storage Device and any minidisks residing on the disk to the existing AIX Operating System hardware and minidisk configurations. It runs the **fsck** command to perform file system consistency checks and mounts the file system if the attributes **vcheck = true** and **mount = true** are present in the corresponding stanza in the **/etc/filesystems** file.

Normally, **varyon** sends only generalized completion messages to standard output, routing more detailed error messages to the file **/etc/varyon.out**.

**Flags**

- c** Does not perform file consistency checking. This flag can be specified only with the **-m** or **-p** flags.
- d hdisknum . . .** Specifies the AIX Operating System name for an IBM 9332 Direct Access Storage Device, where *num* is an integer from 6 to 33, inclusive. The names **hdisk0** through **hdisk5** are reserved for internal fixed drives.

The **varyon** command uses **vrconfig** to configure all minidisks residing on the drive. It uses **fsck** to check each file system whose stanza in the **/etc/filesystems** file contains the attribute **vcheck = true**. It mounts each file system that has the attribute **mount = true**.

- m** *hdnum* . . . Specifies the minidisk or minidisks to be made available for use. **varyon** runs **vrconfig** to configure the minidisk. It configures the disk drive containing each minidisk, if it is not already configured, and checks and mounts each file system.
- p** *directory* . . . Makes the minidisk associated with each specified directory available for use. **varyon** runs **vrconfig** to configure the minidisk. It checks and mounts each file system and configures each disk drive containing the minidisk, if it is not already configured.
- v** Sends all messages to standard output.

## Examples

1. To configure an entire IBM 9332 Direct Access Storage Device:

```
varyon -d hdisk7
```

This configures a disk drive named **hdisk7** into the operating system, configures any minidisks defined on the disk, and performs **fsck** and **mount** functions on file systems as specified by the **/etc/filesystems** file.

2. To configure more than one IBM 9332 Direct Access Storage Device:

```
varyon -d hdisk9 hdisk7 hdisk12
```

This makes disk drives **hdisk9**, **hdisk7**, and **hdisk12** available for use.

3. To configure the device containing a specified minidisk:

```
varyon -m hd8
```

This instructs **varyon** to find the name of the disk drive that contains the minidisk named **hd8**, to configure it into the system, and to add the minidisk definition to the minidisk configuration. **varyon** performs **fsck** and **mount** functions unconditionally on the file system defined on the minidisk.

4. To configure several minidisks:

```
varyon -m hd7 hd8 hd11
```

This makes minidisks **hd7**, **hd8**, and **hd11** available for use, configures the corresponding drives, and unconditionally performs **fsck** and **mount** functions.

5. To configure a minidisk without performing **fsck** functions:

```
varyon -c -m hd7 hd8 hd11
```

6. To configure a minidisk using its mount directory path name:

```
varyon -p /usr/lib
```

This causes the appropriate disk drive and minidisk definitions to be configured into the operating system and unconditionally performs **fsck** and **mount** functions on the file system.

7. To configure several minidisks by specifying their mount directory path names:

```
varyon -p /usr/lib /job/lib
```

8. To avoid **fsck** functions when configuring a minidisk associated with a specified mount directory path name:

```
varyon -c -p /usr/lib
```

9. To direct all detailed status and error messages to standard output:

```
varyon -v -d hdisk10
```

## Files

|                  |                                         |
|------------------|-----------------------------------------|
| /etc/filesystems | Descriptions of mountable file systems. |
| /etc/system      | Default system file.                    |
| /etc/varyon.out  | Default error output file.              |

## Related Information

The following commands: “**fsck**, **dfck**” on page 333, “**mount**” on page 498, and “**vrnconfig**” on page 842.

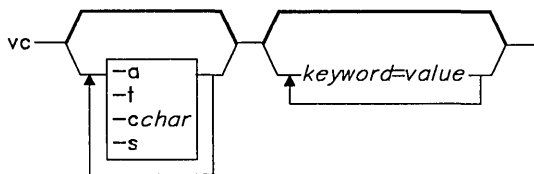
The **filesystems** file in *AIX Operating System Technical Reference*.



## Purpose

Substitutes assigned values in place of keywords.

## Syntax



OL805293

## Description

The `vc` command is used to control writing different versions of a file to standard output. However, since Source Code Control System commands (“**admin**” on page 51, “**get**” on page 359, and “**delta**” on page 236) provide more efficient and complete control, they should be used instead of `vc`.

The `vc` command copies lines from standard input to standard output. The flags and *keywords* on the command line and control statements in the input modify the resulting output. `vc` replaces user declared *keywords* with their *value* assigned on the command line. These *keywords* can be replaced both in text and in control statements.

### Control Statements

A control statement is a single line beginning with a control character (the default control character is a `:` (colon)). They provide conditional processing of the input. The allowable types of control statements are:

**`:if`** *condition*

*text*

**`:end`**

Writes all the lines between the `:if` statement and the matching `:end` to standard output only if *condition* is true. You can nest `:if:end` statements, but once a condition is false, all remaining nested `:if:end` statements are ignored. See “Condition Syntax” on page 827 for the syntax of conditions and allowable operators.

- 
- :del** *keyword* [, *keyword* . . . ]  
Declares *keywords*. All *keywords* must be declared.
- :asg** *keyword*=*value*  
Assigns *value* to *keyword*. An **:asg** statement takes precedence over keyword assignment on the **vc** command line. A later **:asg** statement overrides all earlier assignments of the associated *keyword*. *keywords* declared, but not assigned *values* have null values.
- ::** *text*  
Removes the two leading control characters and replaces *keywords* with their *values*, and then copies the line to the standard output.
- :on**  
**:off**  
Turns on or off *keyword* replacement on all lines.
- :ctl** *char*  
Changes the control character to *char*.
- :msg** *message*  
Writes a message to standard error output in the form:  
Message(*n*): *message*  
The *n* indicates the line number on which the message appeared in the input.
- :err** *message*  
Writes an error message to standard error in the form:  
ERROR: *message*  
ERROR: err statement on line *n* (vc15)  
**vc** stops processing and returns an exit *value* of 1.

## Condition Syntax

| Item          | Statements Allowed                                  |
|---------------|-----------------------------------------------------|
| condition     | ::=or statement<br>::=not or statement              |
| or statement  | ::=and statement<br>::=and statement   or statement |
| and statement | ::=expression<br>::=expression & and statement      |
| expression    | ::=( or statement )<br>::=value operator value      |
| operator      | ::== or != or < or >                                |

| Item value | Statements Allowed |
|------------|--------------------|
|            | ::=ASCII string    |
|            | ::=numeric string  |

The available condition operators and their meanings are as follows:

|            |                                                                                                               |
|------------|---------------------------------------------------------------------------------------------------------------|
| =          | equal                                                                                                         |
| !=         | not equal                                                                                                     |
| &          | and                                                                                                           |
|            | or                                                                                                            |
| >          | greater than                                                                                                  |
| <          | less than                                                                                                     |
| ()         | used for logical groupings                                                                                    |
| <b>not</b> | may only occur immediately after the <i>if</i> , and when present, inverts the value of the entire condition. |

The > and < (greater than and less than symbols) operate only on unsigned integer values; for example: 012 > 12 is false. All other operators take strings as modifiers; for example: 012 != 12 is true. The precedence of the operators, from highest to lowest precedence, is as follows:

```
= != > < all of equal precedence
&
|
```

Parentheses can be used, of course, to alter the order of precedence.

Values must be separated from operators or parentheses by at least one blank or tab.

## Keyword Replacement

A *keyword* must begin and end with the same control character used in control statements. A *keyword* may be up to nine alphanumeric characters, where the first character must be alphabetic. *Keyword values* can be any ASCII string. A numeric *keyword value* is an unsigned string of digits. *values* cannot contain tabs or spaces.

Examples of *keyword=value* assignments are:

```
numlines=4
prog=acctg
pass4=yes
```

The **vc** command removes all control characters and *keywords* from input text lines marked with two control characters as it writes the text to standard output. To prevent a control character from being interpreted, precede it with a backslash, as in the following example:

```
::the :prog: program includes several of the following\:
```

The keyword **:prog:** is replaced by its *value*, but the **\:** is passed to standard output as **:** (colon).

Input lines beginning with a **\** (backslash) followed by a control character are not control lines, and are copied to standard output without the backslash. **vc** writes lines beginning with a backslash and no following control character without any changes (including the initial backslash).

## Flags

- a** Replaces *keywords* surrounded by control characters with their assigned *value* in all text lines (not just those beginning with two control characters).
- cchar** Uses *char* as the control character.
- s** Does not display the warning messages normally displayed to standard error.
- t** Ignores all characters from the beginning of a line up to and including the first tab character for detecting a control statement. If **vc** finds a control character, it ignores all characters up to and including the tab.

## Related Information

The following commands: “**admin**” on page 51, “**delta**” on page 236, and “**get**” on page 359.

# verify

---

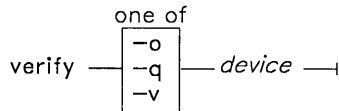
## verify

---

### Purpose

Turns write verification on or off for a particular minidisk.

### Syntax



OL805446

### Description

The **verify** command turns write verification on or off for the specified minidisk *device*. Generally speaking, you can verify any device name that is by shown by the **minidisk** command. More specifically, the name must be a stanza name in the **/etc/system** file.

When write verification is on, the system checks each write operation to the minidisk by comparing the data written to disk with the data in the write buffer. If it detects an uncorrectable error, then it passes an error code back from the write operation.

At system startup, write verification is turned off.

### Flags

- o** Turns write verification off.
- q** Tells you whether write verification is set on or off.
- v** Turns write verification on.

### Files

/dev/config  
/etc/system

## **Related Information**

The following command: “**minidisks**” on page 485.

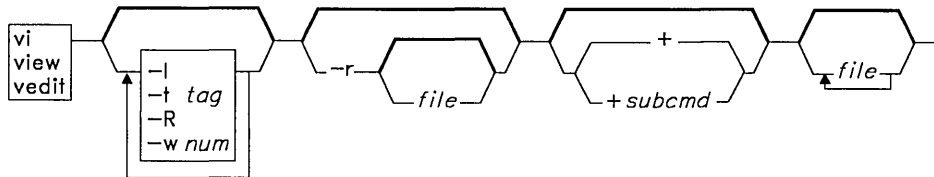
The **mdverify** subroutine in *AIX Operating System Technical Reference*.

## vi, vedit, view

### Purpose

Edits files with a full screen display.

### Syntax



OL805424

### Description

**Warning:** These commands do not support international characters. If you use them to edit a file that contains extended characters, you can lose data.

The `vi` command is a display editor based on an underlying line editor (`ex`). The `view` command is a read-only version of `vi`. The `vedit` command is a version of `vi` intended for beginners. In it, the `report` option is set to 1, and the `showmode` and `novice` options are set. For more information on these options, see “set and map Commands” on page 833.

The `file` parameter specifies the file or files to be edited. If you supply more than one `file` on the command line, `vi` edits each file in the order specified.

When you use `vi`, changes you make to a file are reflected in your display. The position of the cursor on the display indicates its position within the file. The subcommands effect the file at the cursor position.

The `vi` editor has the following operational states:

**Command** This is the initial state. Any subcommand can be entered (except commands that can only be used in the *input* state). When subcommands and the other states end, they return to this state. Pressing ESCAPE (`Esc`) cancels a partial command.

**Input** Entered by the `a`, `A`, `i`, `I`, `o`, `O`, `c`, `C`, `s`, `S`, and `R` subcommands. After entering one of these commands, you can enter text into the file buffer at the current cursor position. To return to the command state, press

ESCAPE (**Esc**) for normal exit or press INTERRUPT (**Alt-Pause**) to end abnormally.

**Last Line** Some subcommands (those with the prefix **:**, **\**, **?**, or **!!**) read input on a line displayed at the bottom of the screen. When you enter the initial character, **vi** places the cursor at the bottom of the screen, where you enter the remaining characters of the command. Press the **Enter** key to perform the subcommand and INTERRUPT (**Alt-Pause**) to cancel it.

## Defining Macros

If you use a sequence of subcommands frequently, you can create a macro that will issue the commands whenever you call the macro. You can enter your sequence of subcommands into buffer *x*. Then You invoke the macro by entering @*x*. Entering @@ repeats the last macro you invoked.

## set and map Commands

The **vi** editor has options which can be changed with the **set** command. To view all the options enter **:set all** while in **vi**. To change the value of an option, enter a line of the form **set option=value** To toggle an option that is either on or off, enter a line of the form **set option** (to set it on) or **set nooption** (to set it off).

All options can be abbreviated when they are set. Some of the most useful options and their abbreviations are:

| Option     | Abbreviation | Description                                                                                                                                             |
|------------|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| autoindent | ai           | Indents automatically (default setting is <b>noai</b> ).                                                                                                |
| autowrite  | aw           | Writes the buffer to the file automatically before the <b>:n</b> , <b>:ta</b> , <b>Ctrl-^</b> , and <b>!</b> subcommands (the default is <b>noaw</b> ). |
| ignorecase | ic           | Ignores case during searches (the default is <b>noic</b> ).                                                                                             |
| lisp       | lisp         | Forces the <b>(</b> , <b>)</b> , <b>{</b> , and <b>}</b> commands to deal with S-expressions (the default is <b>noisp</b> ).                            |
| list       | list         | Prints tabs as Ctrl-I; the end of lines are marked with <b>\$</b> (the default is <b>noisp</b> ).                                                       |
| magic      | magic        | Treats the characters <b>.</b> (dot), <b>[</b> , and <b>*</b> as special characters in scans (the default is <b>magic</b> ).                            |
| number     | nu           | Displays lines prefixed with their line numbers (the default is <b>nonu</b> ).                                                                          |
| paragraphs | para=value   | Defines to <b>vi</b> macro names that start paragraphs (the default is <b>para=IPLPPPQbP LI</b> ).                                                      |
| redraw     | re           | Simulates a smart work station on a dumb work station (the default is <b>nore</b> ).                                                                    |



| Option     | Abbreviation | Description                                                                                                                                                  |
|------------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| sections   | sect =       | Defines to <b>vi</b> macro names that start sections (the default is <b>sect = NHHSH HU</b> ).                                                               |
| shiftwidth | sw =         | Sets the shift distance for the commands <b>&gt;</b> , <b>&lt;</b> , and the input commands <b>Ctrl-D</b> and <b>Ctrl-T</b> (the default is <b>sw = 8</b> ). |
| showmatch  | sm           | Shows the matching opening ( or { as you type the closing ) or } (the default is <b>nosm</b> ).                                                              |
| slowopen   | slow         | Postpones updating the display during inserts (the default is <b>slow</b> ).                                                                                 |
| term       | term =       | Sets the kind of work station you are using (the default is <b>dumb</b> ).                                                                                   |

You can use the **map** command to set a keystroke to a sequence of subcommands. For example, to force @ to delete lines, enter:

```
map @ dd
```

## Flags

- l** Enters **vi** in **LISP** mode. In this mode, **vi** indents appropriately for LISP code and the ( ), { }, [[, and ]] subcommands are modified to act appropriately for LISP.
- r [file]** Recovers *file* after an editor or system crash. If you do not specify a *file* name, **vi** displays a list of all saved files.
- R** Sets the **readonly** option to protect the *file* against overwriting.
- t tag** Edits the file containing the *tag* and positions the editor at its definition.
- wnum** Sets the default window size to *num*. This is useful when you use the editor over a low speed line.
- + [subcmd]** Performs the **ex** subcommand before editing begins. If you do not specify *subcmd*, the cursor is placed on the last line of the file.

## Subcommands

In the following lists, **<ESC>** stands for pressing the ESCAPE key instead of pressing the **Enter** key.

## Counts Before Subcommands

You may prefix many subcommands with a number. `vi` interprets this number in one of the following ways:

1. Go to line *number*:

```
5G
10z
```

2. Go to column *number*:

```
25|
```

3. Scroll *number* lines:

```
10Ctrl-D
10Ctrl-U
```

## Moving Within The File

There are many commands that you can use to move within a file. They can be entered while `vi` is in the **Command** state.

### *Small Movements*

- ← or **h** Moves the cursor one character to the left.
- ↓ or **j** Moves the cursor down one line (but it remains in the same column).
- ↑ or **k** Moves the cursor up one line (but it remains in the same column).
- or **l** Moves the cursor one character to the right.

### *Character Positioning*

- ^** Moves the cursor to the first nonblank character.
- 0** Moves the cursor to the beginning of the line.
- \$** Moves the cursor to the end of the line.
- fx** Moves the cursor to the next *x* character.
- Fx** Moves the cursor to the last *x* character.
- tx** Moves the cursor to one column before the next *x* character.
- Tx** Moves the cursor to one column after the last *x* character.
- ;** Repeat the last **f**, **F**, **t**, or **T** subcommand.

- , Repeat the last **f**, **F**, **t**, or **T** subcommand in the opposition direction.
- num*! Moves the cursor to the specified column.

### ***Words, Sentences, Paragraphs***

- w** Moves the cursor to the next word (includes punctuation as words).
- b** Moves the cursor to the previous word (includes punctuation as words).
- e** Moves the cursor to the end of the word (includes punctuation as words).
- W** Moves the cursor to the next word (ignores punctuation).
- B** Moves the cursor to the previous word (ignores punctuation).
- E** Moves the cursor to the to the end of the word (includes punctuation as part of the current word).

### ***Line Positioning***

- H** Moves the cursor to the top line on the screen.
- L** Moves the cursor to the last line on the screen.
- M** Moves the cursor to the middle line on the screen.
- +** Moves the cursor to the next line at its first nonblank character.
- Moves the cursor to the previous line at its first nonblank character.
- <Enter>** Moves the cursor to the next line at its first nonblank character.

### ***Scrolling***

- Ctrl-U** Scrolls up one half screen.
- Ctrl-D** Scrolls down one half screen.
- Ctrl-F** Scrolls forward one screen.
- Ctrl-B** Scrolls backward one screen.

### ***Searching for Patterns***

- [*num*]G** Places the cursor at line number *num* or to the last line if *num* is not specified.
- /*pattern*** Places the cursor at the next line containing *pattern*.

- 
- ?pattern* Places the cursor at the next previous line containing *pattern*.
  - n* Repeats last search for *pattern* in the same direction.
  - N* Repeats last search for *pattern* in the opposite direction.
  - /pattern/+ num*  
Places the cursor at the *num*th line after the line matching *pattern*.
  - ?pattern?-num*  
Places the cursor at the *num*th line before the line matching *pattern*.
  - %* Finds the parentheses or brace that matches the one at the current cursor position.

### ***Moving to Sentences, Paragraphs, or Sections***

- ]]* Places the cursor at next section (or function if you are in the LISP mode).
- [[* Places the cursor at previous section (or function if you are in the LISP mode).
- (* Places the cursor at the beginning of the previous sentence (or the previous s-expression if you are in the LISP mode).
- )* Places the cursor at the beginning of the next sentence (or the next s-expression if you are in the LISP mode).
- {* Places the cursor at the beginning of the next paragraph (or at the next list if you are in the LISP mode).
- }* Places the cursor at the beginning of the next the paragraph (or at the next list if you are in the LISP mode).

### ***Marking and Returning***

- ``* Moves the cursor to the previous location off current line.
- ''* Moves cursor to the beginning of the line containing the pervious location off the current line.
- mx* Marks the current position with letter *x*.
- `x* Moves cursor to mark *x*.
- 'x* Moves cursor to the beginning of the line containing mark *x*.

### ***Adjusting the Screen***

- Ctrl-L** Clears and redraws the screen.

- Ctrl-R** Redraws the screen and eliminates blank lines marked with a @.
- z** Redraws the screen with the current line top of the screen.
- z-** Redraws the screen with the current line at the bottom of the screen.
- z.** Redraws the screen with the current line at the center of the screen.
- /pattern/z-** Redraws the screen with the line containing *pattern* at the bottom.
- znum.** Makes the window *num* lines long.
- Ctrl-E** Scrolls the window down 1 line.
- Ctrl-Y** Scrolls the window up 1 line.

### Subcommands For Editing

Use the following subcommands to edit your text. Those subcommands that do not have an \* (asterisk) following them enter the *input* state. You return to the **Command** state by pressing the ESCAPE (**Esc**) key. These subcommands affect the text relative to the current cursor position.

#### *Editing the File*

- itext** Inserts *text* before the current cursor position.
- cwtext** Changes word to *text*.
- atext** Inserts *text* after the cursor.
- itext** Inserts *text* before the cursor.
- Atext** Adds *text* to the end of the line.
- Itext** Inserts *text* before the first nonblank character in the line.
- o** Adds an empty line below the current line.
- O** Adds an empty line above the current line.
- rx \*** Replaces the current character with *x*. (Commands followed by \* do not enter the input state.)
- Rtext** Overwrites characters with *text*.
- x \*** Deletes a character.
- X \*** Deletes characters before cursor (**dh**).
- s \*** Substitutes characters (**cl**).

---

|                    |                                                                         |
|--------------------|-------------------------------------------------------------------------|
| <b>S *</b>         | Substitutes lines ( <b>cc</b> ).                                        |
| <b>J *</b>         | Joins lines.                                                            |
| <b>dw *</b>        | Deletes a word.                                                         |
| <b>dd *</b>        | Deletes a line.                                                         |
| <b>D *</b>         | Deletes the rest of the line ( <b>d\$</b> ).                            |
| <b>cw</b>          | Changes a word.                                                         |
| <b>cc</b>          | Changes a line.                                                         |
| <b>C</b>           | Changes rest of line ( <b>c\$</b> ).                                    |
| <b>yw *</b>        | Yanks a word into the <b>undo</b> buffer.                               |
| <b>yy *</b>        | Yanks a line into the <b>undo</b> buffer.                               |
| <b>&lt; &lt; *</b> | Shifts one line to the left.                                            |
| <b>&lt;L</b>       | Shifts all lines from the cursor to the end of the screen to the left.  |
| <b>&gt; &gt; *</b> | Shifts one line to the right.                                           |
| <b>&gt;L</b>       | Shifts all lines from the cursor to the end of the screen to the right. |
| <b>! *</b>         | Indents for LISP.                                                       |
| <b>u *</b>         | Undoes the previous change.                                             |

### *Corrections During Insert*

Use the following commands only while in the **Insert** state. They have different meanings in the **Command** state.

|                                       |                                               |
|---------------------------------------|-----------------------------------------------|
| <b>Ctrl-H</b>                         | Erases last character.                        |
| <b>Ctrl-W</b>                         | Erases last word.                             |
| <b>\</b>                              | Quotes the erase and kill characters.         |
| <b>&lt;ESC&gt;</b>                    | Ends insertion, back to command mode.         |
| <b>Ctrl-?</b>                         | Interrupts, terminates insert or Ctrl-D.      |
| <b>Ctrl-D</b>                         | Goes back to previous <b>autoindent</b> stop. |
| <b>^Ctrl-D (circumflex control-D)</b> | Kills <b>autoindent</b> for this line only.   |
| <b>0Ctrl-D</b>                        | Moves cursor back to left margin.             |
| <b>Ctrl-V</b>                         | Enters nonprinting character.                 |

### *Moving Text*

- p** Puts back text in the **undo** buffer after the cursor.
- P** Puts back text in the **undo** buffer before the cursor.
- "xp** Puts back text from the buffer *x*.
- "xd** Deletes text into the buffer *x*.
- y** Places the object that follows (for example, **w** for word) in the **undo** buffer.
- "xy** Places the object that follows in the *x* buffer, where *x* is any letter.
- Y** Places the line in the **undo** buffer.

### *Restoring and Repeating Changes*

- u** Undoes the last change.
- U** Restores the current line.
- .** Repeats the last change.
- "n p** Retrieves the *n*th last delete.

### **Interrupting, Cancelling, and Exiting vi**

- ZZ** Exits **vi**, saving changes.
- :q** Quits **vi**. If you have changed the buffer contents, **vi** displays a warning message and does not quit.
- :q!** Quits **vi**, discarding changes.
- !!cmd** Executes shell command *cmd* and includes output in buffer.
- :sh** Runs a shell. You can return to **vi** by pressing **Ctrl-D**.
- !:cmd** Runs *cmd*, then returns.
- <ESC>** Ends insert or ends an incomplete subcommand.
- Ctrl-?** Interrupts a subcommand.
- Ctrl-L** Redispays a screen.
- Ctrl-R** Redispays the screen if **Ctrl-L** is the **→** key.

---

## File Manipulation

- :w** Writes the buffer contents to the original file.
- :w *file*** Writes the buffer contents to the named *file*.
- :w! *file*** Overwrites *file* with the buffer contents.
- :e *file*** Edits *file*.
- :e!** Re-edits the current file and discards all changes.
- :e + *file***  
Edits *file* starting at the end.
- :e + *num***  
Edits *file* starting at line *num*.
- :e #** Edits the alternate file. The alternate file is the last file name you entered with a **last line** command.
- :n** Edits next file in the list entered on the command line.
- :n *files*** Specifies new list of files to edit.
- Ctrl-G** Shows current file name and line.
- :ta *tag*** Edits a file containing *tag* at the location of *tag*.

## Related Information

The following commands: “**ed**” on page 280 and “**ex**” on page 312.



# vrconfig

---

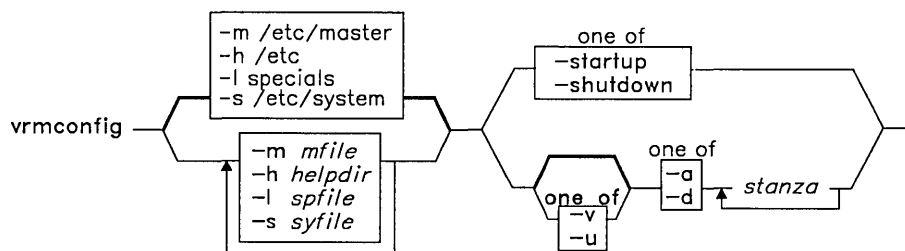
## vrconfig

---

### Purpose

Installs peripheral devices.

### Syntax



OL805400

### Description

The **vrconfig** command installs VRM device drivers. Normally, it runs automatically at each system startup. Its exit value is the number of errors it encountered.

**Note:** Most users will never need to run this command from the command line.

The **vrconfig** command performs its operations through the **/dev/config** driver.

### Flags

- a stanza** Adds devices to a running system. **vrconfig** processes the specified *stanza* in **/etc/system** or the file specified with the **-s** flag.
- d stanza** Deletes a device from a running system. **vrconfig** processes the specified *stanza* in **/etc/system** or the file specified with **-s**. The special file list produced includes commands to remove relevant special files, since **vrconfig** assumes that the device has been removed from the configuration.
- h helpdir** Specifies the directory that contains the configuration helper programs. The default value is **/etc**.
- l spfile** Specifies the output special file list. The default value is **specials**.

- 
- m** *mfile* Specifies the input master configuration file. The default value is **/etc/master**.
  - s** *syfile* Specifies the input system configuration file. The default value is **/etc/system**.
  - u** Causes only AIX-related configuration steps to be performed, that is, AIX driver installation calls. This flag may be used only with the **-a** or **-d** flags.
  - v** Causes only VRM-related configuration steps to be performed, that is, DefineDevice calls. This flag may be used only with the **-a** or **-d** flags.
  - shutdown** Causes all installed drives to be deleted for shutting down the system. Special files are not deleted, since the actual configuration is not considered changed.
  - startup** Causes all defined devices to be configured in at system start up. (Any stanza that contains the attribute **noipl=true** is skipped.) For devices such as minidisks, which remain configured between system restarts, **vrconfig** does not perform “once-only” configuration steps. It does not modify special files that already exist.

## Files

|                          |                                                             |
|--------------------------|-------------------------------------------------------------|
| <b>/etc/configstatus</b> | Status file recording current configuration status.         |
| <b>/etc/system</b>       | Default system file.                                        |
| <b>/etc/master</b>       | Default master file.                                        |
| <b>specials</b>          | Default special file list.                                  |
| <b>/etc</b>              | Default directory containing configuration helper programs. |
| <b>/etc/vrmdd</b>        | Directory containing VRM device driver modules.             |
| <b>/vrm/vrmdd</b>        | Directory containing VRM device driver modules.             |
| <b>/etc/ddi</b>          | Directory containing device specific information files.     |

## Related Information

The following command: “**config**” on page 150.

The **master** and **system** files in *AIX Operating System Technical Reference. Installing and Customizing the AIX Operating System.*

# wait

---

## wait

---

### Purpose

Waits for completion of a process.

### Syntax

wait —|

OL805294

### Description

The **wait** command causes the shell to wait until child processes started with an **&** have completed. The shell waits on all of its currently active children, and the return code from **wait** is zero.

Because the **wait** system call must be run in the parent process, the shell runs **wait** without creating a new process.

**Note:** Not all processes of a three or more stage pipeline are children of the shell, and thus cannot be waited for.

### Example

To wait for all processes started with **&** to finish:

```
wait
```

### Related Information

The following command: “**sh**” on page 637.

# wall

---

## Purpose

Writes a message to all logged-in users.

## Syntax

wall —

OL805017

## Description

The **wall** command reads a message from standard input until it reaches an end-of-file character. It then sends the message to all logged-in users preceded by the following heading:

Broadcast Message from *user*

To override any protections other users have set up, you must be operating with superuser authority. Typically, **the superuser** uses **wall** to warn all users of an impending system shutdown.

**Note:** This command only sends messages to the local node.

## Files

/dev/tty\*

## Related Information

The following commands: “**mesg**” on page 484, “**su**” on page 724, and “**write**” on page 853.

## wc

---

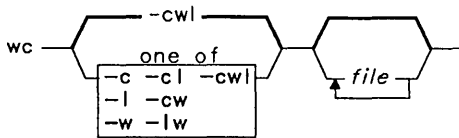
## wc

---

### Purpose

Counts the number of lines, words, and characters in a file.

### Syntax



OL805242

### Description

The **wc** command counts the number of lines, words, or characters in *file* or in the standard input if you do not specify any *files*. It writes the results to standard output. It also keeps a total count for all named files. A word is defined as a string of characters delimited by spaces, tabs, or new-line characters. **wc** counts lines, words, and characters by default.

When you specify more than one *file* on the command line, **wc** displays the name of the file along with the counts.

### Flags

- c Counts bytes.
- l Counts lines.
- w Counts words.

### Examples

1. To display the line, word, and character counts of a file:

```
wc chap1
```

This displays the number of lines, words, and characters in the file chap1.

2. To display only character and word counts:

```
wc -cw chap*
```

This displays the number of characters and words in each file whose name starts with chap, and displays the totals.

# what

---

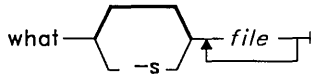
# what

---

## Purpose

Displays identifying information in files.

## Syntax



OL805295

## Description

The **what** command searches the named *files* for all occurrences of the pattern that **get** substitutes for the **%Z%** keyletter (see “Identification Keywords” on page 362). By convention, the value substituted is **@(#)**. **what** writes to standard output whatever follows the pattern up to but not including the first double quotation mark (**"**), greater than symbol (**>**), new-line character **\n**, backslash (**\**), or null character.

The **what** command is intended for use in conjunction with the **get** command, which automatically inserts the identifying information. You can also use **what** on files where the information is inserted manually.

## Flags

**-s** Searches for only the first occurrence of **@(#)**.

## Examples

Suppose that the file `test.c` contains a C program that includes the line:  
`char ident[ ] = "@(#)Test Program";`

If you compile `test.c` to produce `test.o` and `a.out`, then the command:

```
what test.c test.o a.out
```

displays:

```
test.c: Test Program
```

```
test.o: Test Program
```

```
a.out: Test Program
```

## Related Information

The following commands: “**get**” on page 359, and “**help**” on page 391.

The `sccsfile` file in *AIX Operating System Technical Reference*.

The discussion of SCCS in *AIX Operating System Programming Tools and Interfaces*.



# who

---

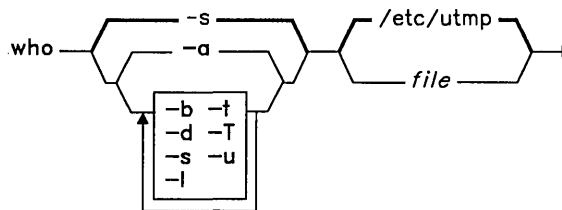
## who

---

### Purpose

Identifies the users currently logged in.

### Syntax



```
who am i —
```

OL805296

### Description

The **who** command with no flags writes to standard output the login name, work station name, and date and time of login for all users currently on the system. **who am i** gives this information only for you.

With flags, **who** can also display the elapsed time since line activity occurred, the process-ID of the command interpreter (shell), logins, logouts, restarts, and changes to the system clock, as well as other processes generated by the **init** process.

The general format of the output of **who** is as follows:

```
name [state] line time activity pid [location] [exit]
```

where:

- **name** is the user's login name.
- **state** indicates whether or not the line is readable by everyone (see the **-T** flag on page 851).
- **line** is the name of the line as found in the directory **/dev**.
- **time** is the time that user logged in.

- **activity** is the hours and minutes since activity last occurred on that user's line. A dot (.) here indicates line activity within the last minute. If the line has been quiet more than 24 hours or has not been used since the last system startup, the entry is marked old.
- **pid** is the process-ID of the user's shell.
- **location** is the location associated with this line as found in file **/etc/ports**. This file can contain information about where the work station is located, the telephone number of the dataset, the type of work station if direct-connected, and other related information.
- **exit** is the exit status of ended processes (see the **-d** flag on page 851).

To obtain its information, **who** normally examines **/etc/utmp**. If you specify another *file*, **who** examines the named *file* instead. This *file* will usually be **/usr/adm/wtmp**, which contains the history of all logins since the file was last created or **/etc/ilog**, which contains the history of invalid logins. Only someone operating with superuser authority or a member of the system group can examine **/etc/ilog**.

**Note:** This command only identifies users on the local node.

## Flags

- a Processes **/etc/utmp** or the named *file* with all flags on.
- b Indicates the time and date of the most recent system startup. The **NLTIME** and **NLLDATE** environment variables control the format of the login time and date.
- d Displays all processes that have expired without being regenerated by **init**. The *exit* field appears for dead processes and contains the termination and exit values (as returned by **wait**) of the dead process. (This flag is useful for determining why a process ended.)
- l Lists only work stations not in use. The *name* field is **LOGIN** in such cases. Other fields are the same as for user entries except that the *state* field doesn't appear.
- s Lists only the *name*, *line*, and *time* fields. (This is the default; thus, **who** and **who -s** are equivalent.) The **NLTIME** environment variable controls the format of the time.
- t Indicates the last change to the system clock by the superuser using the **date** command. The **NLTIME** environment variable controls the format of the time.
- T Displays the *state* of the work station line and indicates who can write to that work station as follows:
  - + writable by anyone
  - writable only by the superuser or its owner
  - ? bad line encountered.

## who

---

- u Displays the user name, work station name, login time, line activity, and process-ID of each current user. The **NLTIME** environment variable controls the format of the login time.

## Examples

1. To display information about who is using the system:

```
who
```

This lists the user name, work station name, and login time of all users currently using the system.

2. To display your user name:

```
who am i
```

This displays the user name you typed when you logged in, the name of the work station you are using, and the time you logged in. Your login user name may be different from your current user name if you have used the **su** command.

3. To display a history of logins, logouts, system startups, and system shutdowns:

```
who /usr/adm/wtmp
```

## Files

```
/etc/utmp  
/usr/adm/wtmp  
/etc/ports
```

## Related Information

The following commands: “**date**” on page 219, “**init**” on page 396, “**login**” on page 453, “**mesg**” on page 484, and “**su**” on page 724.

The **wait** system call and the **ports** and **utmp** files in *AIX Operating System Technical Reference*.

“Overview of International Character Support” in *Managing the AIX Operating System*.

---

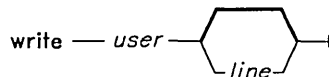
# write

---

## Purpose

Sends messages to other users on the system.

## Syntax



OL805297

## Description

A common use of this command is to **converse** with another logged-in *user*. That is, each user alternately sends and receives short messages from the other work station. Long messages can be sent by first putting the complete message in a file and then redirecting that file as input to the **write** command.

For other *users* to receive your message, they must be logged in, and they must not have refused message permission. When a person you are trying to reach is not logged in, you get the message `user not logged in`. When the person you are trying to reach has refused message permission, you get the message `write: permission denied`.

**Note:** This command only sends messages to users on the local node.

When you run the **write** command, it immediately sends the following message, along with an attention-getting sound (the ASCII BEL character) to the person whose login name you entered.

```
Message from yourid (ttynn)
[date] . . .
```

After successful connection, **write** then sends two ASCII BEL characters to your work station to alert you that whatever you enter now is being sent to the other *user*. Sending continues until you press **Ctrl-D**, at which point **write** sends an end-of-text character to the other work station and exits.

At this point, the other *user* can respond by sending a **write** message back. For this type of exchange, the following convention is useful: When you first **write** to others, wait for them to **write** back before sending any text. End a message with a signal such as `O` (over) to alert the other person to reply. Use `OO` (over and out) when the conversation is finished.

## write

---

When you **write** to a *user* logged in at more than one work station, **write** uses the first login instance found in file `/etc/utmp` as the message delivery point, and you get the message:

```
userid is logged on more than one place.  
You are connected to "work station".  
Other locations are:  
work station
```

You can contact this *user* at another location by specifying the *line*. *line* indicates to which work station (`tty00`, for example) the message should be sent.

Permission to **write** to another *user* is granted or denied by the other *user* with the **mesg** command. Some commands deny message permission while they are running to prevent interference with their output. A user with superuser authority can **write** to any work station regardless of the work station's message permission.

## Examples

1. To write a message to a user who is logged in:

```
write scott  
I need to see you! Meet me  
in my office at 12:30.  
Ctrl-D
```

If your user ID is `janet` and you are using work station `tty3`, then `scott`'s work station displays:

```
Message from janet tty3...  
I need to see you! Meet me  
in my office at 12:30.  
EOF
```

2. To hold a conversation:

```
write scott  
Are you free at 12:30?  
(o)
```

This starts a conversation with `scott`. The `(o)` at the end stands for "over." It tells `scott` that you are waiting for a response. **Do not** press **Ctrl-D** yet because this would end the conversation.

Now `scott` replies by typing:

```
write janet  
No, but I'm free after 3.  
(o)
```

---

And you might respond:

```
write scott
OK. Meet me in my office at 3.
(oo)
```

The (oo) stands for “over and out,” telling Scott that you have nothing more to say. If Scott is also finished (oo), then you both press **Ctrl-D** to end the conversation.

3. To write someone a prepared message:

```
write fred <message.text
```

This writes the contents of the file `message.text` to fred’s work station.

4. To write to the person using a certain work station:

```
write - console
The printer in building 998 has jammed.
Please send help.
Ctrl-D
```

This writes the message to the person logged in at the work station `/dev/console`.

## Files

`/etc/utmp`

## Related Information

The following commands: “**mesg**” on page 484, “**nroff**” on page 525, “**pr**” on page 561, “**sh**” on page 637, “**wall**” on page 845, and “**who**” on page 850.

## wump

---

## wump

---

### Purpose

Plays the game Hunt the Wumpus.

### Syntax

`/usr/games/wump` —

OL805232

### Description

A wumpus is a creature living in a cave with many rooms interconnected by tunnels. You move among the rooms trying to shoot the wumpus with an arrow and trying to avoid being eaten by the wumpus or falling into Bottomless Pits. There are also Super Bats that may pick you up and drop you in some randomly selected room. For moving among the rooms and shooting arrows, **wump** asks appropriate questions and follows your instructions.

After either you kill the wumpus, the wumpus eats you, or you fall into a Bottomless Pit, **wump** asks if you want a new game. To quit the game at any time, press INTERRUPT (Alt-Pause) or END OF FILE (Ctrl-D).

---

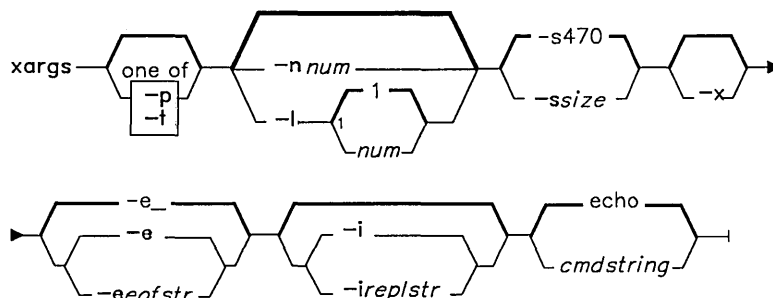
## xargs

---

### Purpose

Constructs argument lists and runs commands.

### Syntax



OL805298

### Description

The **xargs** command runs a command line. It constructs the command line by combining *cmdstring*, a string containing a command and its flags or parameters, with additional arguments read from standard input. It runs *cmdstring* as many times as necessary to process all input arguments. The default *cmdstring* is **echo**.

Arguments read from standard input are character strings delimited by one or more blanks, tabs, or new-line characters. You can embed a blank or a tab in arguments by preceding it with a \ (backslash) or by quoting it. **xargs** reads characters enclosed in single or double quotes as literals and removes the delimiting quotes. It always discards empty lines.

### Flags

- e[*eofstr*]** Sets the logical end-of-file string to *eofstr*. **xargs** reads standard input until it encounters either an end-of-file character or the logical EOF string. If you do not specify the **-e** flag, the default *eofstr* is `_` (the underline character). If you specify **-e** with no *eofstr*, **xargs** interprets the underline character as a literal character rather than as an end-of-file marker.



## xargs

---

- i[replstr]** Takes an entire line as a single argument and inserts it in each instance of *replstr* found in *cmdstring*. A maximum of five arguments in *cmdstring* may each contain one or more instances of *replstr*. **xargs** discards blanks and tabs at the beginning of each line. The argument constructed may not be larger than 255 characters. The default *replstr* is `{}`. This flag also turns on the **-x** flag.
- l[num]** Runs *cmdstring* with the specified *num* of nonempty argument lines read from standard input. The last invocation of *cmdstring* can have fewer argument lines if fewer than *num* remain. A line ends with the first new-line character unless the last character of the line is a blank or a tab. A trailing blank or tab indicates a continuation through the next nonempty line. The default *num* is 1. This flag turns on the **-x** flag.
- nnum** Executes *cmdstring* using as many standard input arguments as possible, up to a maximum of *num*. **xargs** uses fewer arguments if their total size is greater than the number of characters specified by the **-ssize** flag described following. It also uses fewer arguments for the last invocation if fewer than *num* arguments remain. When **-x** is present, each *num* argument must fit the *size* limitation specified by **-x**.
- p** Asks whether or not to run *cmdstring*. It displays the constructed command line, followed by a `? . . .` prompt. Press `y` to run the *cmdstring*. Any other response causes **xargs** to skip that particular invocation of *cmdstring*. You are asked about each invocation.
- ssize** Sets the maximum total size of each argument list. *size* must be a positive integer less than or equal to 470. The default *size* is 470 characters. Note that the character count for *size* includes one extra character for each argument and the number of characters in the command name.
- t** Echoes the *cmdstring* and each constructed argument list to file descriptor 2 (usually standard error).
- x** Stops running **xargs** if any argument list is greater than the number of characters specified by the **-ssize**. This flag is turned on if you specify either the **-i** or **-l** flags. If you do not specify **-i**, **-l**, or **-n**, the total length of all arguments must be within the *size* limit.

**Note:** The **xargs** command ends if it cannot run *cmdstring* or if it receives a return code of -1. When *cmdstring* calls a shell procedure, the shell procedure should explicitly **exit** with an appropriate value to avoid accidentally returning -1. (See “**sh**” on page 637.)

## Examples

1. To use a command on files whose names are listed in a file:

```
xargs lint -a <cfiles1 *
```

If `cfiles` contains the text:

```
main.c readit.c
gettoken.c
putobj.c
```

then `xargs` constructs and runs the command:

```
lint -a main.c readit.c gettoken.c putobj.c
```

Each shell command line can be up to 470 characters long. If `cfiles` contains more file names than fit on a single line, then `xargs` runs the `lint` command with the file names that fit. It then constructs and runs another `lint` command using the remaining file names. Depending on the names listed in `cfiles`, the commands might look like:

```
lint -a main.c readit.c gettoken.c . . .
lint -a getisx.c getprp.c getpid.c . . .
lint -a fltadd.c fltmult.c fltdiv.c . . .
```

This is not quite the same as running `lint` once with all the file names. The `lint` command checks cross-references between files. However, in this example it cannot check between `main.c` and `fltadd.c`, or between any two files listed on separate command lines.

For this reason you may want to run the command only if all the file names fit on one line. Tell `xargs` this by using the `-x` flag:

```
xargs -x lint -a <cfiles
```

If all the file names in `cfiles` do not fit on one command line, then `xargs` displays an error message.

2. To construct commands that contain a certain number of file names:

```
xargs -t -n2 diff <<end
starting chap1 concepts chap2 writing
chap3
end
```

This constructs and runs `diff` commands that contain two file names each (`-n2`):

```
diff starting chap1
diff concepts chap2
diff writing chap3
```

## xargs

---

The `-t` flag tells `xargs` to display each command before running it so that you can see what is happening. The `<<end` and `end` define a “Here Document,” which uses the text entered before the `end` line as standard input for the `xargs` command. For more details, see “Inline Input Documents” on page 650.

3. To insert file names into the middle of commands:

```
ls | xargs -t -i mv {} {}.old
```

This renames all files in the current directory by adding `.old` to the end of each name. The `-i` tells `xargs` to insert each line of the `ls` directory listing where a `{}` appears. If the current directory contains the files `chap1`, `chap2`, and `chap3`, then this constructs the commands:

```
mv chap1 chap1.old
mv chap2 chap2.old
mv chap3 chap3.old
```

4. To run a command on files that you select individually:

```
ls | xargs -p -n1 ar r lib.a
```

This allows you to select files to add to the library `lib.a`. The `-p` flag tells `xargs` to display each `ar` command it constructs and ask if you want to run it. Type `y` and press **Enter** to run the command. Press **Enter** alone if you do not want to run it.

## Related Information

The following command: “**sh**” on page 637.

---

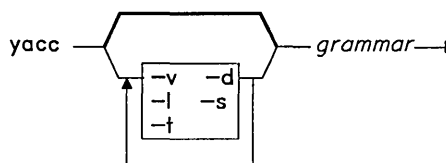
## yacc

---

### Purpose

Generates a LR(1) parsing program from input consisting of a context-free grammar specification.

### Syntax



OL805300

### Description

The **yacc** command converts a context-free grammar into a set of tables for a simple automaton that executes an LR(1) parsing algorithm. The grammar can be ambiguous; specified precedence rules are used to break ambiguities.

You must compile the output file, **y.tab.c**, with a C Language compiler to produce a function **yyparse**. This function must be loaded with the lexical analyzer function **yylex**, as well as **main** and **yyerror**, an error-handling routine (you must provide these routines). The **lex** command is useful for creating lexical analyzers usable by **yacc**.

For more detailed discussion of **yacc** and its operations, see *AIX Operating System Programming Tools and Interfaces*.

### Flags

- d Produces the file **y.tab.h**. This contains the **#define** statements that associate the **yacc**-assigned token codes with your token names. This allows source files other than **y.tab.c** to access the token codes by including this header file.
- l Does not include any **#line** constructs in **y.tab.c**. Use this only after the grammar and associated actions are fully debugged.
- s Breaks the **yyparse** function into several smaller functions. Since its size is somewhat proportional to that of the grammar, it is possible for **yyparse** to become too large to compile, optimize, or execute efficiently.

## yacc

---

- t Compiles run-time debugging code. By default, this code is not included when **y.tab.c** is compiled. However, the run-time debugging code is under the control of **YYDEBUG**, a global variable for the **cc** command preprocessor. If **YYDEBUG** has a nonzero value, the C compiler (**cc**) includes the debugging code, whether or not the **-t** flag was used. Without compiling this code, **yyparse** will have a faster operating speed.
- v Prepares the file **y.output**. It contains a readable description of the parsing tables and a report on conflicts generated by grammar ambiguities.

## Files

|                  |                                  |
|------------------|----------------------------------|
| y.output         |                                  |
| y.tab.c          |                                  |
| y.tab.h          | Definitions for token names.     |
| yacc.tmp,        |                                  |
| yacc.debug       | Temporary file.                  |
| yacc.acts        | Temporary file.                  |
| /usr/lib/yaccpar | Parser prototype for C programs. |

## Related Information

The following command: “**lex**” on page 432.

The description of **yacc** in *AIX Operating System Programming Tools and Interfaces*.

---

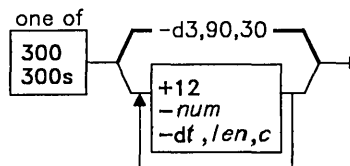
## 300

---

### Purpose

Handles special line-motion functions for DASI 300/300s work stations.

### Syntax



OL805193

### Description

**Note:** If your work station has a **PLOT** switch, make sure this switch is turned on before using this command.

The **300** command reads standard input, processes its input for printing on the DASI 300, GSI 300, or DTC 300 work stations, and writes to standard output. The **300s** command performs the same functions for the DASI 300s, GSI 300s, and DTC 300s. They convert the input files' motion control characters for half-line forward, half-line reverse, and full-line reverse into motion commands recognized by these work stations.

You can use the **300** and **300s** commands to draw Greek characters and other special symbols that require more than one vertical line, and it allows you to use 12-pitch text. For a discussion of special symbols and greek characters supported by **300**, see "**greek**" on page 379.

The **nroff** command can be used with the **300** command to format text. **300** must be used if you use special delays or formatting options. You can either pipe from **nroff** to **300** or use the **-T300** flag with **nroff** to specify the *printing device*. The movement control of the **300** command usually produces better aligned output than **nroff -T300**.

When using **nroff**, the **-s** flag or **.rd** requests are required for inserting paper manually or changing fonts in the middle of a document. In these cases, you must press the line feed key to continue printing.

Using the **300** command with the **neqn** command will give you the best display of your equations. You can use the following sequence to display equations:

```
neqn file . . . | nroff | 300
```

**Note:** Some special characters cannot be correctly printed in column 1 because the print head cannot be moved to the left from that position.

If your output contains Greek characters or reverse line feeds, use a friction-feed platen instead of a forms tractor. A forms tractor slips when reversing direction.

## Flags

**-dt,len,c** Controls output delay factors. The default setting is **-d3,90,30**. DASI 300 is too slow to handle very long lines, too many tab characters, or long strings with no blanks and no identical characters. One null character is inserted in a line for every set of *t* tabs, and for every contiguous string of *c* nonblank, nontab characters. When a line is longer than *len* bytes, several nulls (the line length divided by 20, plus one) are inserted at the end of that line. In all three cases, the nulls delay the output enough to avoid a problem. Items can be omitted from the end of the list, implying the default values. Entering zero for *t* results in insertion of two null bytes per tab, while entering zero for *c* results in insertion of two null bytes per character.

When printing C Language programs, using **-d0,1** will help adjust for the many indentation levels. When printing files like **/etc/passwd**, using **-d3,30,5** will help print it properly.

This flag affects carriage return and line feed delays. The **stty** parameters **nl0 cr2** or **nl0 cr3** are recommended for most uses.

**-num** Controls the size of half-line spacing. The default half-line values (which are exact half-lines) of *num* are:

10-pitch, 6 lines-per-inch, num = 4  
12-pitch, 8 lines-per-inch, num = 3  
12-pitch, 6 lines-per-inch, num = 4

You can use other values for *num* to change the appearance of subscripts and superscripts. For example, **-2** makes **nroff** half-lines act like quarter-lines.

**+12** Uses 12-pitch, 6 lines-per-inch text. The DASI 300 normally allows only two combinations: 10-pitch, 6 lines per inch, or 12-pitch, 8 lines per inch. To use the 12-pitch, 6 lines-per-inch combination, set the PITCH switch to 12 and use the **+12** flag on the command line.

## Related Information

The following commands: “**450**” on page 866, “**eqn, neqn, checkeq**” on page 300, “**graph**” on page 375, “**mesg**” on page 484, “**nroff**” on page 525, “**stty**” on page 717, “**tbl**” on page 739, and “**tplot**” on page 762.

The **greek** miscellaneous facility in *AIX Operating System Technical Reference*.

---

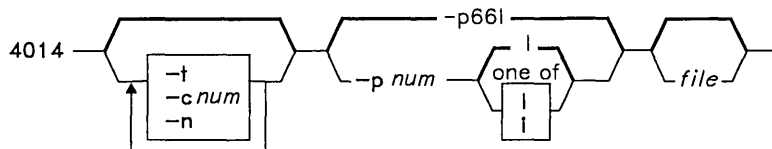
## 4014

---

### Purpose

Formats a full page 66-line screen display for a Tektronix 4014 work station.

### Syntax



OL805195

### Description

The **4014** command reads a *file* (standard input by default) and writes a 66-line page display to standard output. It also divides the screen into a specified number of columns, adding an eight space page offset when it uses the default single column format. It interprets tabs, spaces, backspaces, and TELETYPE Model 37 half-line and reverse-line sequences correctly. At the end of each page, **4014** waits for a line feed from the keyboard before continuing. While **4014** is waiting, you can send commands to the shell by entering **!AIX-cmd**, where *AIX-cmd* is a AIX command.

### Flags

- cnum** Divides the screen into *num* columns and waits after the last column. The default is a single, full page-width column.
- n** Starts displaying at the current cursor position and does not erase the screen.
- pnuml**
- pnumi** Sets page length to *num* lines (l, the default) or to *num* inches (i).
- t** Does not wait between pages.

### Related Information

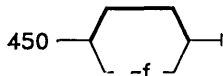
The following commands: “**pr**” on page 561, “**tc**” on page 742, and “**troff**” on page 526.



## Purpose

Handles special line-motion functions for the DASI 450 work station

## Syntax



```
450 ———— |
           \-f- /
```

OL805194

## Description

The **450** command reads standard input, processes its data for output on a DASI 450 or an equivalent work station (such as the DIABLO 1620 or XEROX 1700). It converts half-line forward, half-line reverse, and full-line reverse motions to the correct vertical motions on standard output. It attempts to draw Greek characters and other special symbols in the same manner as the **300** command vertical line space. See “**greek**” on page 379 for a list of symbols supported by **450**.

Use **450** with the **nroff -s** flag or **.rd** requests when you need to insert paper manually or change fonts in the middle of a document. Instead of using the return key in these cases, you must use the line feed key to get any response. In many cases you can use **nroff -T450** instead of the **450** command. However, you must use **450** if you require special delays or options. In a few cases, using **450** may produce better aligned output. You can pipe the output of the **neqn** command to **450** to print equations neatly.

**Note:** Make sure the **PLOT** switch is turned on before using this command. Also, the **SPACING** switch should be in the desired position, either 10- or 12-pitch. For either setting, vertical spacing is 6 lines per inch unless changed to 8 lines per inch by an escape sequence.

Some special characters cannot be correctly printed in column 1 because the print head cannot be moved to the left from that position.

If your output contains Greek characters or reverse linefeeds, use a friction-feed platen instead of a forms tractor. A forms tractor tends to slip when reversing direction.

## Flag

- f Permits the use of ETX/ACK protocol with 1200 bps printers. You cannot use **450** with this flag in a pipeline or if you redirect its output. Instead it must drive the printer directly.

## Related Information

The following commands: “**300**” on page 863, “**eqn, neqn, checked**” on page 300, “**graph**” on page 375, “**greek**” on page 379, “**mesg**” on page 484, “**nroff**” on page 525, “**stty**” on page 717, “**tabs**” on page 729, “**tbl**” on page 739, “**tplot**” on page 762, and “**troff**” on page 526.

The **greek** miscellaneous facility in *AIX Operating System Technical Reference*.



---

## Appendix A. AIX Device Table

| Special File      | Description                             |
|-------------------|-----------------------------------------|
| appltrace         | Application trace pseudo device driver  |
| config            | Configuration pseudo device driver      |
| console           | Console device                          |
| error             | Error-logging interface                 |
| fd[ <i>num</i> ]  | Diskette drive, block device            |
| fp                | Floating-point function                 |
| hd[ <i>num</i> ]  | Fixed disk drive, block device          |
| hft               | High function terminal                  |
| kmem              | Kernel memory image                     |
| lp[ <i>num</i> ]  | Line printer                            |
| mem               | Memory image                            |
| null              | The null device                         |
| nvrnm             | Non-volatile memory image               |
| osm               | System message interface                |
| prf               | AIX Operating System profiler           |
| rfd[ <i>num</i> ] | Diskette drive, raw device              |
| rhd[ <i>num</i> ] | Fixed disk drive, raw device            |
| rmt[ <i>num</i> ] | Streaming tape                          |
| termio            | General terminal interface              |
| tty[ <i>num</i> ] | Controlling terminal interface          |
| unixtrace         | Kernel trace event pseudo device driver |
| vrntrace          | VRM trace event pseudo device driver    |

---

## Appendix B. Program Cross-Reference Index

| <b>Command</b> | <b>Program</b>           | <b>Command</b> | <b>Program</b>           |
|----------------|--------------------------|----------------|--------------------------|
| acctcms        | Multi-User Services      | bfs            | Extended Services        |
| acctcom        | Multi-User Services      | bj             | Extended Services        |
| acctcon1       | Multi-User Services      | bs             | Extended Services        |
| acctcon2       | Multi-User Services      | cal            | Extended Services        |
| acctdisk       | Multi-User Services      | calendar       | Extended Services        |
| acctdusg       | Multi-User Services      | cat            | AIX Operating System     |
| acctmerg       | Multi-User Services      | cb             | AIX Operating System     |
| accton         | Multi-User Services      | cc             | AIX Operating System     |
| acctprc1       | Multi-User Services      | cd             | AIX Operating System     |
| acctprc2       | Multi-User Services      | cdc            | Extended Services        |
| acctwtmp       | Multi-User Services      | cflow          | Extended Services        |
| actman         | AIX Operating System     | chargefee      | Multi-User Services      |
| adb            | Extended Services        | checkcw        | Extended Services        |
| adduser        | AIX Operating System     | checkeq        | Extended Services        |
| admin          | Extended Services        | checkmm        | Extended Services        |
| ar             | AIX Operating System     | chgrp          | AIX Operating System     |
| arithmetic     | Extended Services        | chmod          | AIX Operating System     |
| as             | AIX Operating System     | chown          | AIX Operating System     |
| at             | AIX Operating System     | chparm         | AIX Operating System     |
| awk            | AIX Operating System     | chparm         | Installation/Maintenance |
| back           | Extended Services        | chroot         | Extended Services        |
| backup         | AIX Operating System     | ckpacct        | Multi-User Services      |
| backup         | Installation/Maintenance | ckprereq       | AIX Operating System     |
| banner         | Extended Services        | clri           | AIX Operating System     |
| basename       | AIX Operating System     | cmp            | AIX Operating System     |
| batch          | AIX Operating System     | col            | Extended Services        |
| bc             | Extended Services        | comb           | Extended Services        |
| bdiff          | Extended Services        | comm           | Extended Services        |

| Command  | Program                  | Command   | Program              |
|----------|--------------------------|-----------|----------------------|
| compress | Extended Services        | deroff    | Extended Services    |
| confer   | Multi-User Services      | devices   | AIX Operating System |
| config   | AIX Operating System     | devnm     | AIX Operating System |
| connect  | AIX Operating System     | df        | AIX Operating System |
| copy     | AIX Operating System     | dfsc      | AIX Operating System |
| cp       | AIX Operating System     | di        | AIX Operating System |
| cpio     | AIX Operating System     | diff      | AIX Operating System |
| cpp      | AIX Operating System     | diff3     | Extended Services    |
| craps    | Extended Services        | diffmk    | Extended Services    |
| crash    | AIX Operating System     | dircmp    | Extended Services    |
| cron     | AIX Operating System     | dirname   | AIX Operating System |
| crontab  | AIX Operating System     | diskug    | Multi-User Services  |
| csh      | Extended Services        | display   | AIX Operating System |
| csplit   | Extended Services        | dodisk    | Multi-User Services  |
| ctab     | Extended Services        | dos       | AIX Operating System |
| ctag     | AIX Operating System     | dosdel    | AIX Operating System |
| cut      | AIX Operating System     | dosdir    | AIX Operating System |
| cvid     | AIX Operating System     | dosread   | AIX Operating System |
| cw       | Extended Services        | doswrite  | AIX Operating System |
| cxref    | Extended Services        | dsipc     | Distributed Services |
| date     | AIX Operating System     | dsstate   | Distributed Services |
| dc       | Extended Services        | dsldxprof | Distributed Services |
| dcopy    | Extended Services        | dsxlate   | Distributed Services |
| dd       | AIX Operating System     | du        | AIX Operating System |
| dd       | Installation/Maintenance | dump      | Extended Services    |
| defkey   | AIX Operating System     | dumpfmt   | AIX Operating System |
| del      | AIX Operating System     | echo      | AIX Operating System |
| delete   | AIX Operating System     | ed        | AIX Operating System |



| <b>Command</b> | <b>Program</b>           | <b>Command</b> | <b>Program</b>           |
|----------------|--------------------------|----------------|--------------------------|
| delta          | Extended Services        | fuser          | Extended Services        |
| ed             | Installation/Maintenance | fwtmp          | Multi-User Services      |
| edit           | Extended Services        | get            | Extended Services        |
| egrep          | AIX Operating System     | getopt         | AIX Operating System     |
| env            | AIX Operating System     | gettext        | AIX Operating System     |
| eqn            | Extended Services        | getty          | AIX Operating System     |
| errdead        | AIX Operating System     | graph          | Extended Services        |
| errdemon       | AIX Operating System     | graph          | Multi-User Services      |
| errpd          | AIX Operating System     | greek          | Multi-User Services      |
| errpt          | AIX Operating System     | grep           | Extended Services        |
| errstop        | AIX Operating System     | groups         | AIX Operating System     |
| errupdate      | AIX Operating System     | grpck          | Extended Services        |
| ex             | Extended Services        | hangman        | Extended Services        |
| expr           | AIX Operating System     | hashcheck      | Extended Services        |
| factor         | Extended Services        | hashmake       | Extended Services        |
| false          | AIX Operating System     | help           | Extended Services        |
| ff             | Extended Services        | hp             | Multi-User Services      |
| fgrep          | AIX Operating System     | hyphen         | Extended Services        |
| file           | AIX Operating System     | id             | Multi-User Services      |
| find           | AIX Operating System     | init           | AIX Operating System     |
| fish           | Extended Services        | init           | Installation/Maintenance |
| format         | AIX Operating System     | install        | AIX Operating System     |
| format         | Installation/Maintenance | installp       | AIX Operating System     |
| fortune        | Extended Services        | inudocm        | AIX Operating System     |
| fsck           | AIX Operating System     | inurecv        | AIX Operating System     |
| fsck           | Installation/Maintenance | inurest        | AIX Operating System     |
| fsdb           | AIX Operating System     | inusave        | AIX Operating System     |
| fsdb           | Installation/Maintenance | inuumsg        | AIX Operating System     |

| <b>Command</b> | <b>Program</b>           | <b>Command</b> | <b>Program</b>           |
|----------------|--------------------------|----------------|--------------------------|
| inuupdt        | AIX Operating System     | mdrc           | AIX Operating System     |
| ipcrm          | AIX Operating System     | mant           | Extended Services        |
| ipcs           | AIX Operating System     | mesg           | Multi-User Services      |
| istat          | Extended Services        | minidisks      | AIX Operating System     |
| join           | Extended Services        | mkdir          | AIX Operating System     |
| joinconf       | Multi-User Services      | mkfs           | AIX Operating System     |
| keyboard       | AIX Operating System     | mknod          | AIX Operating System     |
| kill           | AIX Operating System     | mkfs           | Installation/Maintenance |
| killall        | AIX Operating System     | mknod          | Installation/Maintenance |
| lastlogin      | Multi-User Services      | mm             | Extended Services        |
| ld             | AIX Operating System     | mmt            | Extended Services        |
| lex            | Extended Services        | monacct        | Multi-User Services      |
| li             | AIX Operating System     | moo            | Extended Services        |
| line           | AIX Operating System     | mount          | AIX Operating System     |
| link           | AIX Operating System     | mount          | Installation/Maintenance |
| lint           | Extended Services        | move           | AIX Operating System     |
| ln             | AIX Operating System     | mv             | AIX Operating System     |
| locator        | AIX Operating System     | mvdir          | AIX Operating System     |
| login          | AIX Operating System     | mvmd           | AIX Operating System     |
| logname        | AIX Operating System     | mvt            | Extended Services        |
| lorder         | AIX Operating System     | ncheck         | AIX Operating System     |
| lp             | AIX Operating System     | neqn           | Extended Services        |
| ls             | AIX Operating System     | newform        | Extended Services        |
| ls             | Installation/Maintenance | newgrp         | AIX Operating System     |
| m4             | AIX Operating System     | news           | AIX Operating System     |
| mail           | AIX Operating System     | nice           | AIX Operating System     |
| make           | AIX Operating System     | nl             | Extended Services        |
| makekey        | AIX Operating System     | nm             | AIX Operating System     |

| <b>Command</b> | <b>Program</b>       | <b>Command</b> | <b>Program</b>           |
|----------------|----------------------|----------------|--------------------------|
| nohup          | AIX Operating System | ps             | AIX Operating System     |
| nroff          | Extended Services    | ptx            | Extended Services        |
| nulladm        | Multi-User Services  | puttext        | AIX Operating System     |
| number         | Extended Services    | pwck           | Extended Services        |
| od             | Extended Services    | pwd            | AIX Operating System     |
| open           | AIX Operating System | qdaemon        | AIX Operating System     |
| pack           | Extended Services    | quiz           | Extended Services        |
| passwd         | AIX Operating System | rc             | AIX Operating System     |
| paste          | Extended Services    | regcmp         | Extended Services        |
| pcat           | Extended Services    | remove         | Multi-User Services      |
| pdisable       | AIX Operating System | restore        | AIX Operating System     |
| penable        | AIX Operating System | restore        | Installation/Maintenance |
| pg             | AIX Operating System | rm             | AIX Operating System     |
| phold          | AIX Operating System | rm             | Installation/Maintenance |
| piobe          | AIX Operating System | rmail          | AIX Operating System     |
| pr             | AIX Operating System | rmdel          | Extended Services        |
| prctmp         | Multi-User Services  | rmdir          | AIX Operating System     |
| prdaily        | Multi-User Services  | rmdir          | Installation/Maintenance |
| prfdc          | Extended Services    | rsh            | AIX Operating System     |
| prfld          | Extended Services    | runacct        | Multi-User Services      |
| prfpr          | Extended Services    | sa1            | Multi-User Services      |
| prfsnap        | Extended Services    | sa2            | Multi-User Services      |
| prfstat        | Extended Services    | sact           | Extended Services        |
| print          | AIX Operating System | sadc           | Multi-User Services      |
| prof           | Extended Services    | sag            | Multi-User Services      |
| proto          | Extended Services    | sar            | Multi-User Services      |
| prs            | Extended Services    | sccsdiff       | Extended Services        |
| prtacct        | Multi-User Services  | sdb            | Extended Services        |

| Command   | Program                  | Command   | Program                  |
|-----------|--------------------------|-----------|--------------------------|
| sdiff     | Extended Services        | sync      | Installation/Maintenance |
| sed       | AIX Operating System     | su        | AIX Operating System     |
| setmnt    | AIX Operating System     | sum       | AIX Operating System     |
| sh        | AIX Operating System     | tab       | Extended Services        |
| sh        | Installation/Maintenance | tabs      | Extended Services        |
| shlib     | AIX Operating System     | tail      | Extended Services        |
| shutacct  | Multi-User Services      | tapechk   | AIX Operating System     |
| shutdown  | AIX Operating System     | tar       | Extended Services        |
| size      | AIX Operating System     | tbl       | Extended Services        |
| skulker   | AIX Operating System     | tc        | Multi-User Services      |
| sleep     | AIX Operating System     | tctl      | AIX Operating System     |
| sno       | Extended Services        | tctl      | Installation/Maintenance |
| sort      | AIX Operating System     | tee       | AIX Operating System     |
| sound     | AIX Operating System     | termdef   | AIX Operating System     |
| spell     | Extended Services        | test      | AIX Operating System     |
| spellhist | Extended Services        | tic       | Extended Services        |
| spellin   | Extended Services        | time      | AIX Operating System     |
| spellprog | Extended Services        | timex     | AIX Operating System     |
| spline    | Multi-User Services      | timex     | Multi-User Services      |
| split     | AIX Operating System     | touch     | AIX Operating System     |
| splp      | AIX Operating System     | tplot     | Multi-User Services      |
| startup   | Multi-User Services      | tput      | Extended Services        |
| stat      | Multi-User Services      | tr        | Extended Services        |
| strip     | AIX Operating System     | trace     | AIX Operating System     |
| STTY      | AIX Operating System     | trcrpt    | AIX Operating System     |
| stty      | AIX Operating System     | trcstop   | AIX Operating System     |
| stty      | Installation/Maintenance | trcupdate | AIX Operating System     |
| sync      | AIX Operating System     | troff     | Extended Services        |

| <b>Command</b> | <b>Program</b>           | <b>Command</b> | <b>Program</b>           |
|----------------|--------------------------|----------------|--------------------------|
| true           | AIX Operating System     | uusub          | Extended Services        |
| tsort          | AIX Operating System     | uuto           | Extended Services        |
| ttt            | Extended Services        | uux            | Extended Services        |
| tty            | AIX Operating System     | val            | Extended Services        |
| turnacct       | Multi-User Services      | varyon         | AIX Operating System     |
| turnoff        | Extended Services        | vc             | Extended Services        |
| turnon         | Extended Services        | vedit          | Extended Services        |
| umask          | AIX Operating System     | verify         | AIX Operating System     |
| umount         | AIX Operating System     | vi             | Extended Services        |
| umount         | Installation/Maintenance | view           | Extended Services        |
| uname          | AIX Operating System     | vrmdir         | AIX Operating System     |
| unget          | Extended Services        | vrmdir         | Installation/Maintenance |
| uniq           | Extended Services        | wait           | AIX Operating System     |
| units          | Extended Services        | wall           | AIX Operating System     |
| unlink         | AIX Operating System     | wc             | AIX Operating System     |
| unmount        | AIX Operating System     | what           | AIX Operating System     |
| unpack         | Extended Services        | who            | AIX Operating System     |
| untab          | Extended Services        | write          | AIX Operating System     |
| updatep        | AIX Operating System     | wtmpfix        | Multi-User Services      |
| users          | AIX Operating System     | wump           | Extended Services        |
| uuclean        | Extended Services        | xargs          | AIX Operating System     |
| uucp           | Extended Services        | yacc           | Extended Services        |
| uulog          | Extended Services        | 300s           | Multi-User Services      |
| uuname         | Extended Services        | 4014           | Multi-User Services      |
| uupick         | Extended Services        | 450            | Multi-User Services      |
| uustat         | Extended Services        |                |                          |

---

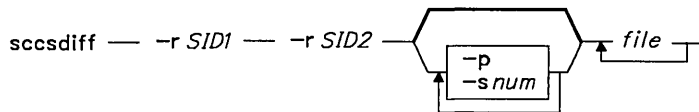
## **Appendix C. Details on Reading Syntax Diagrams**

---

## File Input

Some commands must read a file as their input, some must read standard input, and some can read both. The syntax diagrams show you which case applies to each particular command.

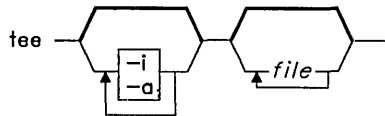
If a command must read a file as its input, the diagram shows a path through the parameter representing the file. It will have no branch around it. The following is an example of a command that must read a file:



OL805258

Since the only path from the command name to the end passes through *file*, you must supply a file name to the `sccsdiff` command.

When a command can only read standard input, you will find no place in the diagram to supply a file name. For example:



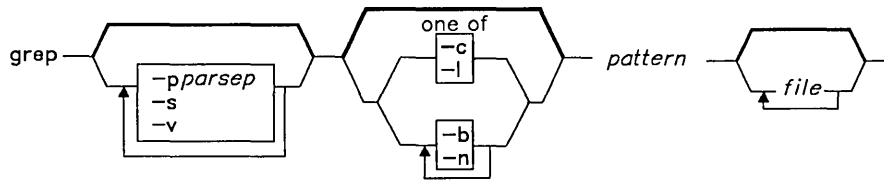
OL805272

The *file* in the diagram refers to an output file. But since there is no place for the input to be specified as a file, you can assume that the `tee` command reads standard input only. Therefore, if you want to use the `tee` command, you must supply its input through a pipeline, through redirection of the output from another command, or directly from the keyboard (if it is standard input).

Most commands can either read standard input or files for their input. The diagrams show this by branching and giving you a choice of entering a file name or nothing. The default action is to read standard input if no file name is supplied (if this is not the default action, it is explained under “Description” or with a footnote to the diagram).

---

The following shows how this is diagramed:



OL805375

When you use the **grep** command, you can give it a specific file name, and it will read that file for its input. If you do not give it a file name, it will read standard input. The following are valid command lines for **grep**:

```
grep AAA
grep -sAAA
grep AAA memo
grep -s -v AAA memo
grep AAA memo letter report
grep -s AAA memo letter report
```

In the first two cases, **grep** reads the standard input.

**Note:** Some commands require that you enter a - (minus) when you want the command to read standard input. If that is the case, it is explained under "Description", NOT in the diagram. Usually this is done so that you can read several files as input and you can include standard input as one of the "files".

## Syntax Diagrams

### Command Only

The simplest syntax diagram shows a command that is entered literally on the command line with nothing else. Some examples are:

logname —|

devices —|

OL805145

OL805306

The blue command name means it should be entered literally. As you follow the line away from the command name, the next item you encounter is the end mark, which indicates nothing more can be entered with that command. The correct way to enter these commands is to simply enter:



---

logname  
devices

## Commands with Required Parameters

Many diagrams have parameters in italics to represent specific values that you must enter on the command line. When entering the command, you replace the parameter with the value you need. One example is:

```
unlink — file —|
```

OL805227

To read the diagram, you start with the command name in blue type. As you move along the line to the right, you reach the parameter *file*, and you must supply a file name. As you move further to the right, you reach the end mark, and must stop.

Suppose you want to unlink three files named **report**, **memo**, and **letter**. You would have to enter the **unlink** command three times:

```
unlink report  
unlink memo  
unlink letter
```

Note that you must enter a file after the command. The diagram shows that you do not have the choice of entering the command without one.

Often, you may want to enter more than one parameter on the command line. If you are allowed to do so, the diagrams show it with a *repeat arrow*, an arrow that provides a path back to an earlier part of the diagram. For example:

```
sact — file —|
```

```
echo — string —|
```

OL805063

OL805115

With all these diagrams, you start with the command name, then you reach the parameter and must supply a value. Then you have a choice; you can continue to the right to the end mark and end your command, or you can follow the repeat arrow around to the point between the command name and the parameter, and enter another parameter. All the following are valid command lines according to these diagrams:

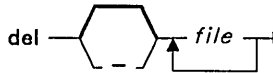
```
sact s.letter  
sact s.letter s.memo s.report  
echo hello!  
echo hello there!
```

---

If there is a maximum number of parameters that you can enter, the diagram tells you that number. If no maximum is mentioned, then you can enter as many parameters as you wish, within the length of your command line.

## Commands with Optional Flags or Parameters

Many commands have optional flags or parameters. If something is optional, you have a choice of paths in the diagram. One takes you around the flags, and the other takes you through them. For example:

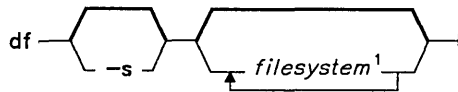


OL805049

With this diagram, as you move to the right from the command name, you reach a branch. You can either take the upper branch to the *file* parameter, or you can take the lower branch through the flag to the end mark. The dark line around the flag is a default line. It shows that this is the path taken if you enter nothing from the lower branch. There are only two ways to enter this command, according to this diagram. They are

```
del file1 file2 file3
del - file1 file2 file3
```

As the command syntax becomes more complicated, the features of the diagrams are combined to help you enter commands properly. The next diagram shows a command that accepts an optional flag and an optional parameter that can be repeated.



---

<sup>1</sup>The default action is to provide information for each file system in `/etc/filesystems` with the attribute `free=true`.

OL805052

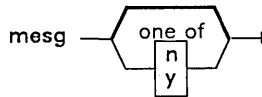
Once you have typed the command name, you have a choice of entering the flag. Then you can stop, you can supply one file-system name, or you can supply more than one if you follow the repeat arrow instead of proceeding to the end mark. Several valid command lines are:

```
df
df -s
df system1
df -s system1
df system1 system2 system3
df -s system1 system2 system3
```

---

## Commands Taking Only One Flag or Parameter

Many commands have flags or parameters that cannot or should not be entered together on the command line. When this is the case, the mutually exclusive items are enclosed in a single-choice box. The following diagram shows a command with optional flags that are mutually exclusive:



OL805036

There are only three valid ways to enter this command:

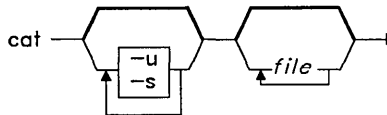
```
msg
msg n
msg y
```

Whenever you see a box with several flags or parameters in it and see above the box the words "one of", you can only choose one item in the box.

## Commands That Can Take Several Flags

With many commands, you can enter as many of a group of flags or parameters as you want. If this is the case, the items are in a box that has a repeat arrow around it. You can follow the arrow around and through the box until you have selected all the items you want to use. Note that as you continue to go through the box, you should not normally choose an item that was previously chosen. Some commands do not work if you choose flags more than once.

The following is an example of a diagram with flags in a box surrounded by a repeat arrow.



OL805086

With `cat`, you can enter only the command name by following the default line over the box. You can enter one flag and then continue to the end mark. You can also follow the arrow around and choose both flags. The following are valid command lines:

```
cat
cat -u
cat -s
cat -u -s
```

---

The following is *not* a valid command line:

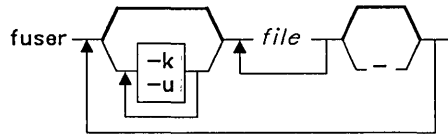
```
cat -u -s -u
```

Once you have chosen an item from the box, you cannot choose it again, unless a footnote tells you otherwise.

## Commands That Can Repeat Part of a Sequence

Some commands allow you to choose flags for each parameter that they read. When this is the case, another repeat arrow allows you to go back to an earlier part of the diagram.

One example is:



OL805055

In this diagram, there are three repeat arrows. The first allows you to choose one or both flags. The second allows you to have **fuser** read more than one file. The third allows you to repeat the complete sequence from the beginning of the diagram to the end. The following are all correct ways to enter **fuser** on the command line:

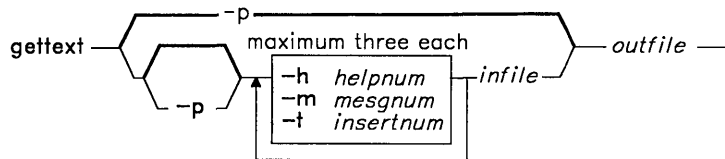
```
fuser memo
fuser memo -
fuser -k memo
fuser -k -u memo
fuser -k -u memo letter
fuser -k -u memo -
fuser -k memo - -u -k letter -u report -
```

The third arrow does allow you to enter the same flag repeatedly, but only **after** at least one file name has been entered. If you follow the diagram, you cannot repeat a flag without entering at least one file name after it.

---

## Commands With Default Values

The default line can show more than just an alternate path around flags and parameters. Sometimes, a flag is set by default or a parameter has a default value. When that is the case, the default value is shown in the normal font in the default line. For example:



OL805130

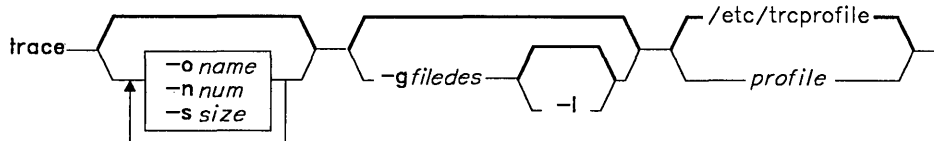
If you do not enter any flags with `gettext`, the `-p` flag is set by default. Selecting the `-h`, `-m`, or `-t` flags will turn the `-p` flag off. The following two command lines are equivalent ways of entering `gettext`:

```
gettext -p report
gettext report
```

The following are valid command lines using the non-default flags for `gettext`:

```
gettext -h2 report
gettext -m3 report
gettext -m3 memo report
```

You can have default parameter values and default flags. The following is an example:



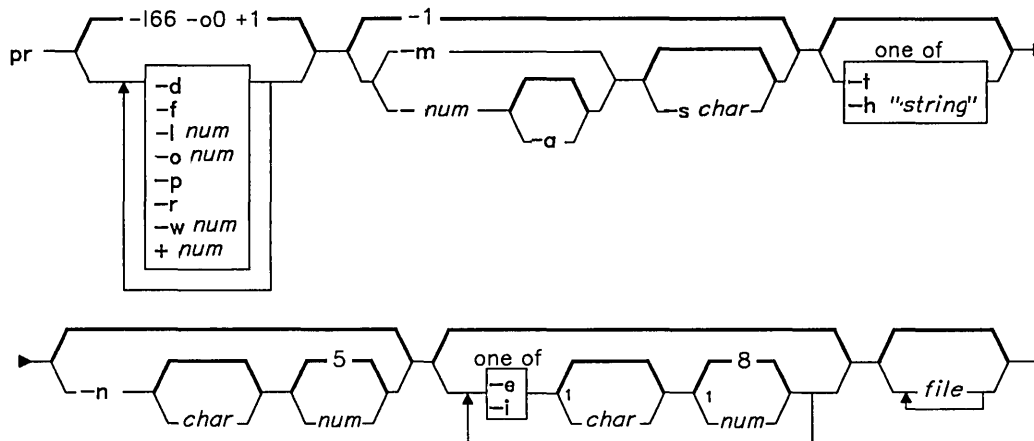
OL805279

If no `profile` file is supplied, `trace` reads the file `/etc/trcprofile`. The following are equivalent command lines:

```
trace
trace /etc/trcprofile
```

## Diagrams Taking More Than One Line

Some of the more complex diagrams do not fit on one line. They are marked with an arrowhead where they break, and they continue on the next line. For example:



<sup>1</sup> Do not put a blank between these items.

OL805437

As you follow this diagram, you can choose as many as eight flags. Then, you have a choice of entering no flag, the `-m` flag, or the `-` flag with a number, `num`. On any of these three branches, you continue until you reach the arrowhead. Then you must go down to the next arrowhead, which is right below the command name, `pr`. You can then choose no flag, the `-t` flag, or the `-h` flag. These can be followed by the `-n` flag, and its parameters. You then reach another arrowhead, and you must go down another line to the second arrowhead beneath the command name. As you work your way through this line, you finally reach the end mark. Note that while following the diagram will impose a specific order to the flags, you do not need to strictly follow that order when entering the command. If strict order is important, it is stated under "Description" in the commands discussion. The following are some of the ways you can enter `pr` on the command line:

```
pr
pr -d
pr -o4 -r -m -sX memo letter
pr -r -m -t -n4 -iX3 memo letter report
pr -m -n4 -r -iX3 -t memo report letter
pr -l30 5 -3 -a -nX -iX3 -eY memo report
```

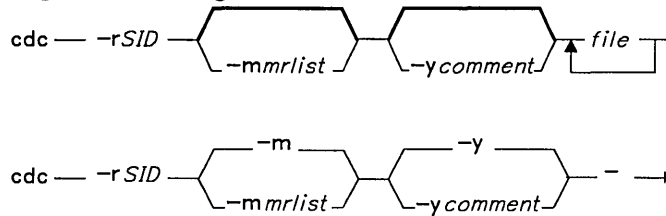
There are a few other features to note in this diagram. One is that many flags that take parameters have default values. Entering `-n` is equivalent to entering `-n5`, since `5` is `num`'s default value. This diagram also has a footnote. Footnotes are used to show

---

information that cannot be diagramed. In this case, it tells you that you cannot put a space between the `-e` or `-i` flags and their parameters. You cannot type `-e Y` or `-i X 3`. You have to type them without spaces, according to the footnote.

## Commands With More Than One Diagram

Several commands are shown with two or more diagrams. For example, the `cdc` command requires two diagrams:



OL805088

When you want `cdc` to read named files, you look at the first diagram for the syntax. It shows that the `-m` and `-y` flags are optional, but they must have parameters. However, if you want `cdc` to read standard input, you look at the second diagram. It shows that you must supply a `-` (hyphen) to read standard input. It also shows that the `-m` and `-y` flags are required, while their parameters are optional. The following are ways you can enter `cdc` according to the diagrams:

```
cdc -r15.2 s.memo s.letter
cdc -r15.2 -mfile1 s.letter
cdc -r15.2 -m -y -
cdc -r15.2 -mfile1 -y -
```

By following the above rules, you can enter commands using the correct syntax.

---

## Figures

|                                                |     |
|------------------------------------------------|-----|
| 1. SCCS Header Flags .....                     | 54  |
| 2. SID Determination .....                     | 363 |
| 3. Delta Table Keywords .....                  | 575 |
| 4. Header Flag Keywords .....                  | 576 |
| 5. Other Keywords .....                        | 577 |
| 6. <b>tbl</b> Column and Item Specifiers ..... | 740 |
| 7. Configuration File Parameters .....         | 803 |





---

## Glossary

**access.** To obtain data from or put data in storage.

**access permission.** A group of designations that determine who can access a particular AIX file and how the user may access the file.

**account.** The log in directory and other information that give a user access to the system.

**activity manager.** A collection of system-supplied tasks allowing users to manage their activities. Provides the ability to list current activities (Activity List) and to begin, cancel, hide, and activate activities.

**All Points Addressable (APA) display.** A display that allows each pel to be individually addressed. An APA display allows for images to be displayed that are not made up of images predefined in character boxes. Contrast with *character display*.

**allocate.** To assign a resource, such as a disk file or a diskette file, to perform a specific task.

**alphabetic.** Pertaining to a set of letters a through z.

**alphanumeric character.** Consisting of letters, numbers and often other symbols, such as punctuation marks and mathematical symbols.

**American National Standard Code for Information Interchange (ASCII).** The code developed by ANSI for information interchange among data processing systems, data communications systems, and associated equipment. The ASCII character set consists of 7-bit control characters and symbolic characters.

**American National Standards Institute.** An organization sponsored by the Computer and Business Equipment Manufacturers Association for establishing voluntary industry standards.

**application.** A program or group of programs that apply to a particular business area, such as the Inventory Control or the Accounts Receivable application.

**application program.** A program used to perform an application or part of an application.

**argument.** Numbers, letters, or words that change the way a command works.

**ASCII.** See *American National Standard Code for Information Interchange*.

**attribute.** A characteristic. For example, the attribute for a displayed field could be blinking.

**auto carrier return.** The system function that places carrier returns automatically within the text and on the display. This is accomplished by moving whole words that exceed the line end zone to the next line.

**backend.** The program that sends output to a particular device. There are two types of backends: friendly and unfriendly.

**background process.** (1) A process that does not require operator intervention that can be run by the computer while the work station is used to do other work. (2) A mode of program execution in which the shell does not wait for program completion before prompting the user for another command.

**backup copy.** A copy, usually of a file or group of files, that is kept in case the original

---

file or files are unintentionally changed or destroyed.

**backup diskette.** A diskette containing information copied from a fixed disk or from another diskette. It is used in case the original information becomes unusable.

**bad block.** A portion of a disk that can never be used reliably.

**base address.** The beginning address for resolving symbolic references to locations in storage.

**base name.** The last element to the right of a full path name. A filename specified without its parent directories.

**batch printing.** Queuing one or more documents to print as a separate job. The operator can type or revise additional documents at the same time. This is a background process.

**batch processing.** A processing method in which a program or programs process records with little or no operator action. This is a background process. Contrast with *interactive processing*.

**binary.** (1) Pertaining to a system of numbers to the base two; the binary digits are 0 and 1. (2) Involving a choice of two conditions, such as on-off or yes-no.

**bit.** Either of the binary digits 0 or 1 used in computers to store information. See also *byte*.

**block.** (1) A group of records that is recorded or processed as a unit. Same as *physical record*. (2) In data communications, a group of records that is recorded, processed, or sent as a unit. (3) A block is 512 bytes long. (4) A logical block is 2048 bytes long.

**block file.** A file listing the usage of blocks on a disk.

**block special file.** A special file that provides access to an input or output device is capable of

supporting a file system. See also *character special file*.

**bootstrap.** A small program that loads larger programs during system initialization.

**branch.** In a computer program an instruction that selects one of two or more alternative sets of instructions. A conditional branch occurs only when a specified condition is met.

**breakpoint.** A place in a computer program, usually specified by an instruction, where execution may be interrupted by external intervention or by a monitor program.

**buffer.** (1) A temporary storage unit, especially one that accepts information at one rate and delivers it at another rate. (2) An area of storage, temporarily reserved for performing input or output, into which data is read, or from which data is written.

**burst pages.** On continuous-form paper, pages of output that can be separated at the perforations.

**byte.** The amount of storage required to represent one character; a byte is 8 bits.

**call.** (1) To activate a program or procedure at its entry point. Compare with *load*.

**callouts.** An AIX kernel parameter establishing the maximum number of scheduled activities that can be pending simultaneously.

**cancel.** To end a task before it is completed.

**carrier return.** (1) In text data, the action causing line ending formatting to be performed at the current cursor location followed by a line advance of the cursor. Equivalent to the carriage return of a typewriter. (2) A keystroke generally indicating the end of a command line.

**case sensitive.** Able to distinguish between uppercase and lowercase letters.

**character.** A letter, digit, or other symbol.

---

**character display.** A display that uses a character generator to display predefined character boxes of images (characters) on the screen. This kind of display cannot address the screen any less than one character box at a time. Contrast with *All Points Addressable display*.

**character key.** A keyboard key that allows the user to enter the character shown on the key. Compare with *function keys*.

**character position.** On a display, each location that a character or symbol can occupy.

**character set.** A group of characters used for a specific reason; for example, the set of characters a printer can print or a keyboard can support.

**character special file.** A special file that provides access to an input or output device. The character interface is used for devices that do not use block I/O. See also *block special file*.

**character string.** A sequence of consecutive characters.

**character variable.** The name of a character data item whose value may be assigned or changed while the program is running.

**child.** (1) Pertaining to a secured resource, either a file or library, that uses the user list of a parent resource. A child resource can have only one parent resource. (2) In the AIX Operating System, child is a *process* spawned by a parent process that shares resources of parent process. Contrast with *parent*.

**C language.** A general-purpose programming language that is the primary language of the AIX Operating System.

**class.** Pertaining to the I/O characteristics of a device. AIX devices are classified as block or character.

**close.** (1) To end an activity and remove that window from the display.

**code.** (1) Instructions for the computer. (2) To write instructions for the computer; to *program*. (3) A representation of a condition, such as an error code.

**code segment.** See *segment*.

**collating sequence.** The sequence in which characters are ordered within the computer for sorting, combining, or comparing.

**color display.** A display device capable of displaying more than two colors and the shades produced via the two colors, as opposed to a monochrome display.

**column.** A vertical arrangement of text or numbers.

**column headings.** Text appearing near the top of columns of data for the purpose of identifying or titling.

**command.** A request to perform an operation or run a program. When parameters, arguments, flags, or other operands are associated with a command, the resulting character string is a single command.

**command interpreter.** A program that sends instructions to the kernel; also called an interface.

**command line.** The area of the screen where commands are displayed as they are typed.

**command line editing keys.** Keys for editing the command line.

**command programming language.** Facility that allows programming by the combination of commands rather than by writing statements in a conventional programming language.

**compile.** (1) To translate a program written in a high-level programming language into a machine language program. (2) The computer actions required to transform a source file into an executable object file.

---

**compress.** (1) To move files and libraries together on disk to create one continuous area of unused space. (2) In data communications, to delete a series of duplicate characters in a character string.

**concatenate.** (1) To link together. (2) To join two character strings.

**condition.** An expression in a program or procedure that can be evaluated to a value of either true or false when the program or procedure is running.

**configuration.** The group of machines, devices, and programs that make up a computer system. See also *system customization*.

**configuration file.** A file that specifies the characteristics of a system or subsystem, for example, the AIX queueing system.

**consistent.** Pertaining to a file system, without internal discrepancies.

**console.** (1) The main AIX display station. (2) A device name associated with the main AIX display station.

**constant.** A data item with a value that does not change. Contrast with *variable*.

**context search.** A search through a file whose target is a character string.

**control block.** A storage area used by a program to hold control information.

**control commands.** Commands that allow conditional or looping logic flow in shell procedures.

**control program.** Part of the AIX Operating System system that determines the order in which basic functions should be performed.

**controlled cancel.** The system action that ends the job step being run, and saves any new data already created. The job that is running can continue with the next job step.

**copy.** The action by which the user makes a whole or partial duplicate of already existing data.

**crash.** An unexpected interruption of computer service, usually due to a serious hardware or software malfunction.

**current directory.** The directory that is active, and can be displayed with the **pwd** command.

**current line.** The line on which the cursor is located.

**current working directory.** See *current directory*.

**cursor.** (1) A movable symbol (such as an underline) on a display, used to indicate to the operator where the next typed character will be placed or where the next action will be directed. (2) A marker that indicates the current data access location within a file.

**cursor movement keys.** The directional keys used to move the cursor.

**customize.** To describe (to the system) the devices, programs, users, and user defaults for a particular data processing system.

**cylinder.** All fixed disk or diskette tracks that can be read or written without moving the disk drive or diskette drive read/write mechanism.

**daemon.** See *daemon process*.

**daemon process.** A process begun by the root or the root shell that can be stopped only by the root. Daemon processes generally provide services that must be available at all times such as sending data to a printer.

**data block.** See *block*.

**data communications.** The transmission of data between computers, or remote devices or both (usually over long distance).

---

**data stream.** All information (data and control information) transmitted over a data link.

**debug.** (1) To detect, locate, and correct mistakes in a program. (2) To find the cause of problems detected in software.

**default.** A value that is used when no alternative is specified by the operator.

**default directory.** The directory name supplied by the operating system if none is specified.

**default drive.** The drive name supplied by the operating system if none is specified.

**default value.** A value stored in the system that is used when no other value is specified.

**delete.** To remove. For example, to delete a file.

**dependent work station.** A work station having little or no standalone capability, that must be connected to a host or server in order to provide any meaningful capability to the user.

**device.** An electrical or electronic machine that is designed for a specific purpose and that attaches to your computer, for example, a printer, plotter, disk drive, and so forth.

**device driver.** A program that operates a specific device, such as a printer, disk drive, or display.

**device name.** A name reserved by the system that refers to a specific device.

**diagnostic.** Pertaining to the detection and isolation of an error.

**diagnostic aid.** A tool (procedure, program, reference manual) used to detect and isolate a device or program malfunction or error.

**diagnostic routine.** A computer program that recognizes, locates, and explains either a fault

in equipment or a mistake in a computer program.

**digit.** Any of the numerals from 0 through 9.

**directory.** A type of file containing the names and controlling information for other files or other directories.

**disable.** To make nonfunctional.

**discipline.** Pertaining to the order in which requests are serviced, for example, first-come-first-served (fcfs) or shortest job next (sjn).

**disk I/O.** Fixed-disk input and output.

**diskette.** A thin, flexible magnetic plate that is permanently sealed in a protective cover. It can be used to store information copies from the disk or another diskette.

**diskette drive.** The mechanism used to read and write information on diskettes.

**display device.** An output unit that gives a visual representation of data.

**display screen.** The part of the display device that displays information visually.

**display station.** A device that includes a keyboard from which an operator can send information to the system and a display screen on which an operator can see the information sent to or received from the computer.

**dump.** (1) To copy the contents of all or part of storage, usually to an output device. (2) Data that has been dumped.

**dump diskette.** A diskette that contains a dump or is prepared to receive a dump.

**dump formatter.** Program for analyzing a dump.

**EBCDIC.** See *extended binary-coded decimal interchange code*.

---

**EBCDIC character.** Any one of the symbols included in the 8-bit EBCDIC set.

**edit.** To modify the form or format of data.

**edit buffer.** A temporary storage area used by an editor.

**editor.** A program used to enter and modify programs, text, and other types of documents and data.

**emulation.** Imitation; for example, when one computer imitates the characteristics of another computer.

**enable.** To make functional.

**enter.** To send information to the computer by pressing the **Enter** key.

**entry.** A single input operation on a work station.

**environment.** The settings for shell variables and paths set associated with each process. These variables can be modified later by the user.

**error-correct backspace.** An editing key that performs editing based on a cursor position; the cursor is moved one position toward the beginning of the line, the character at the new cursor location is deleted, and all characters following the cursor are moved one position toward the beginning of the line (to fill the vacancy left by the deleted element).

**escape character.** A character that suppresses the special meaning of one or more characters that follow.

**exit value.** A numeric value that a command returns to indicate whether it completed successfully. Some commands return exit values that give other information, such as whether a file exists. Shell programs can test exit values to control branching and looping.

**expression.** A representation of a value. For example, variables and constants appearing alone or in combination with operators.

**extended binary-coded decimal interchange code (EBCDIC).** A set of 256 eight-bit characters.

**feature.** A programming or hardware option, usually available at an extra cost.

**field.** (1) An area in a record or panel used to contain a particular category of data. (2) The smallest component of a record that can be referred to by a name.

**FIFO.** See *first-in-first-out*.

**file.** A collection of related data that is stored and retrieved by an assigned name.

**file name.** The name used by a program to identify a file. See also *label*.

**filename.** In DOS, that portion of the file name that precedes the extension.

**file specification (filespec).** The name and location of a file. A file specification consists of a drive specifier, a path name, and a file name.

**file system.** The collection of files and file management structures on a physical or logical mass storage device, such as a diskette or minidisk.

**filetab.** An AIX kernel parameter establishing the maximum number of files that can be open simultaneously.

**filter.** A command that reads standard input data, modifies the data, and sends it to standard output.

**first-in-first-out (FIFO).** A named permanent pipe. A FIFO allows two unrelated processes to exchange information using a pipe connection.

**fixed disk.** A flat, circular, nonremoveable plate with a magnetizable surface layer on

---

which data can be stored by magnetic recording.

**fixed-disk drive.** The mechanism used to read and write information on fixed disk.

**flag.** A modifier that appears on a command line with the command name that defines the action of the command. Flags in the AIX Operating System almost always are preceded by a dash.

**font.** A family or assortment of characters of a given size and style.

**foreground.** A mode of program execution in which the shell waits for the program specified on the command line to complete before returning your prompt.

**format.** (1) A defined arrangement of such things as characters, fields, and lines, usually used for displays, printouts, or files. (2) The pattern which determines how data is recorded.

**formatted diskette.** A diskette on which control information for a particular computer system has been written but which may or may not contain any data.

**free list.** A list of available space on each file system. This is sometimes called the free-block list.

**free-block list.** See *free list*.

**full path name.** The name of any directory or file expressed as a string of directories and files beginning with the root directory.

**function.** A synonym for procedure. The C language treats a function as a data type that contains executable code and returns a single value to the calling function.

**function keys.** Keys that request actions but do not display or print characters. Included are the keys that normally produce a printed character, but when used with the code key produce a function instead. Compare with *character key*.

**generation.** For some remote systems, the translation of configuration information into machine language.

**Gid.** See *group number*.

**global.** Pertains to information available to more than one program or subroutine.

**global action.** An action having general applicability, independent of the context established by any task.

**global character.** The special characters \* and ? that can be used in a file specification to match one or more characters. For example, placing a ? in a file specification means any character can be in that position.

**global search.** The process of having the system look through a document for specific characters, words, or groups of characters.

**global variable.** A symbol defined in one program module, but used in other independently assembled program modules.

**graphic character.** A character that can be displayed or printed.

**group name.** A name that uniquely identifies a group of users to the system.

**group number (Gid).** A unique number assigned to a group of related users. The group number can often be substituted in commands that take a group name as an argument.

**hardware.** The equipment, as opposed to the programming, of a computer system.

**header.** Constant text that is formatted to be in the top margin of one or more pages.

**header label.** A special set of records on a diskette describing the contents of the diskette.

**here document.** Data contained within a shell program or procedure (also called *inline input*).



---

**highlight.** To emphasize an area on the display by any of several methods, such as brightening the area or reversing the color of characters within the area.

**history file.** A file containing a log of system actions and operator responses.

**hog factor.** In system accounting, an analysis of how many times each command was run, how much processor time and memory it used, and how intensive that use was.

**home directory.** (1) A directory associated with an individual user. (2) The user's current directory on login or after issuing the **cd** command with no argument.

**I/O.** See *input/output*.

**ID.** Identification.

**IF expressions.** Expressions within a procedure, used to test for a condition.

**indirect block.** A block containing pointers to other blocks. Indirect blocks can be single-indirect, double-indirect, or triple-indirect.

**informational message.** A message providing information to the operator, that does not require a response.

**initial program load (IPL).** The process of loading the system programs and preparing the system to run jobs. See *initialize*.

**initialize.** To set counters, switches, addresses, or contents of storage to zero or other starting values at the beginning of, or at prescribed points in, the operation of a computer routine.

**inline input.** See *here document*.

**i-node.** The internal structure for managing files in the system. I-nodes contain all of the information pertaining to the node, type, owner, and location of a file. A table of i-nodes is stored near the beginning of a file system.

**i-number.** A number specifying a particular i-node on a file system.

**inodetab.** An AIX kernel parameter that establishes a table in memory for storing copies of i-nodes for all active files.

**input.** Data to be processed.

**input device.** Physical devices used to provide data to a computer.

**input file.** A file opened by a program so that the program can read from that file.

**input list.** A list of variables to which values are assigned from input data.

**input redirection.** The specification of an input source other than the standard one.

**input-output file.** A file opened for input and output use.

**input-output device number.** A value assigned to a device driver by the guest operating system or to the virtual device by the virtual resource manager. This number uniquely identifies the device regardless of whether it is real or virtual.

**input/output (I/O).** Pertaining to either input, output, or both between a computer and a device.

**interactive processing.** A processing method in which each system user action causes response from the program or the system. Contrast with *batch processing*.

**interface.** A shared boundary between two or more entities. An interface might be a hardware component to link two devices together or it might be a portion of storage or registers accessed by two or more computer programs.

**interleave factor.** Specification of the ratio between contiguous physical blocks (on a fixed-disk) and logically contiguous blocks (as in a file).

---

**interrupt.** (1) To temporarily stop a process. (2) In data communications, to take an action at a receiving station that causes the sending station to end a transmission. (3) A signal sent by an I/O device to the processor when an error has occurred or when assistance is needed to complete I/O. An interrupt usually suspends execution of the currently executing program.

**IPL.** See *initial program load*.

**job.** (1) A unit of work to be done by a system. (2) One or more related procedures or programs grouped into a procedure.

**job queue.** A list, on disk, of jobs waiting to be processed by the system.

**justify.** To print a document with even right and left margins.

**kbuffers.** An AIX kernel parameter establishing the number of buffers that can be used by the kernel.

**K-byte.** See *kilobyte*.

**kernel.** The memory-resident part of the AIX Operating System containing functions needed immediately and frequently. The kernel supervises the input and output, manages and controls the hardware, and schedules the user processes for execution.

**kernel parameters.** Variables that specify how the kernel allocates certain system resources.

**key pad.** A physical grouping of keys on a keyboard (for example, numeric key pad, and cursor key pad).

**keyboard.** An input device consisting of various keys allowing the user to input data, control cursor and pointer locations, and to control the dialog between the user and the display station

**keylock feature.** A security feature in which a lock and key can be used to restrict the use of the display station.

**keyword.** One of the predefined words of a programming language; a reserved word.

**keyword argument.** One type of variable assignment that can be made on the command line.

**kill.** An AIX Operating System command that stops a process.

**kill character.** The character that is used to delete a line of characters entered after the user's prompt.

**kilobyte.** 1024 bytes.

**kprocs.** An AIX kernel parameter establishing the maximum number of processes that the kernel can run simultaneously.

**label.** (1) The name in the disk or diskette volume table of contents that identifies a file. See also *file name*. (2) The field of an instruction that assigns a symbolic name to the location at which the instruction begins, or such a symbolic name.

**left margin.** The area on a page between the left paper edge and the leftmost character position on the page.

**left-adjust.** The process of aligning lines of text at the left margin or at a tab setting such that the leftmost character in the line or field is in the leftmost position. Contrast with *right-adjust*.

**library.** A collection of functions, calls, subroutines, or other data.

**licensed program product (LPP).** Software programs that remain the property of the manufacturer, for which customers pay a license fee.

**line editor.** An editor that modifies the contents of a file one line at a time.

**linefeed.** An ASCII character that causes an output device to move forward one line.

---

**link.** A connection between an i-node and one or more file names associated with it.

**literal.** A symbol or a quantity in a source program that is itself data, rather than a reference to data.

**load.** (1) To move data or programs into storage. (2) To place a diskette into a diskette drive, or a magazine into a diskette magazine drive. (3) To insert paper into a printer.

**loader.** A program that reads run files into main storage, thus preparing them for execution.

**local.** Pertaining to a device directly connected to your system without the use of a communications line. Contrast with *remote*.

**log.** To record; for example, to log all messages on the system printer. A list of this type is called a log, such as an error log.

**log in.** To begin a session at a display station.

**log in shell.** The program, or command interpreter, started for a user at log in.

**log off.** To end a session at a display station.

**log out.** To end a session at a display station.

**logical device.** A file for conducting input or output with a physical device.

**loop.** A sequence of instructions performed repeatedly until an ending condition is reached.

**main storage.** The part of the processing unit where programs are run.

**maintenance system.** A special version of the AIX Operating System which is loaded from diskette and used to perform system management tasks.

**major device number.** A system identification number for each device or type of device.

**mapped files.** Files on the fixed-disk that are accessed as if they are in memory.

**mask.** A pattern of characters that controls the keeping, deleting, or testing of portions of another pattern of characters.

**matrix.** An array arranged in rows and columns.

**maxprocs.** An AIX kernel parameter establishing the maximum number of processes that can be run simultaneously by a user.

**memory.** Storage on electronic chips. Examples of memory are random access memory, read only memory, or registers. See *storage*.

**menu.** A displayed list of items from which an operator can make a selection.

**message.** (1) A response from the system to inform the operator of a condition which may affect further processing of a current program. (2) Information sent from one user in a multi-user operating system to another.

**minidisk.** A logical division of a fixed disk.

**minor device number.** A number used to specify various types of information about a particular device, for example, to distinguish among several printers of the same type.

**mode word.** An i-node field that describes the type and state of the i-node.

**modem.** See *modulator-demodulator*.

**modulation.** Changing the frequency or size of one signal by using the frequency or size of another signal.

**modulator-demodulator (modem).** A device that converts data from the computer to a signal that can be transmitted on a communications line, and converts the signal received to data for the computer.

**module.** (1) A discrete programming unit that usually performs a specific task or set of tasks. Modules are subroutines and calling programs

---

that are assembled separately, then linked to make a complete program. (2) See *load module*.

**mount.** To make a file system accessible.

**mountab.** An AIX kernel parameter establishing the maximum number of file systems that can be mounted simultaneously.

**multiprogramming.** The processing of two or more programs at the same time.

**multivolume file.** A diskette file occupying more than one diskette.

**nest.** To incorporate a structure or structures of some kind into a structure of the same kind. For example, to nest one loop (the nested loop) within another loop (the nesting loop); to nest one subroutine (the nested subroutine) within another subroutine (the nesting subroutine).

**network.** A collection of products connected by communication lines for information exchange between locations.

**new-line character.** A control character that causes the print or display position to move to the first position on the next line.

**null.** Having no value, containing nothing.

**null character (NUL).** The character hex 00, used to represent the absence of a printed or displayed character.

**numeric.** Pertaining to any of the digits 0 through 9.

**object code.** Machine-executable instruction, usually generated by a compiler from source code written in a higher level language. Consists of directly executable machine code. For programs that must be linked, object code consists of relocatable machine code.

**octal.** A base eight numbering system.

**open.** (1) To make a file available to a program for processing.

**operating system.** Software that controls the running of programs; in addition, an operating system may provide services such as resource allocation, scheduling, input/output control, and data management.

**operation.** A specific action (such as move, add, multiply, load) that the computer performs when requested.

**operator.** A symbol representing an operation to be done.

**output.** The result of processing data.

**output devices.** Physical devices used by a computer to present data to a user.

**output file.** A file that is opened by a program so that the program can write to that file.

**output redirection.** The specification of an output destination other than the standard one.

**override.** (1) A parameter or value that replaces a previous parameter or value. (2) To replace a parameter or value.

**overwrite.** To write output into a storage or file space that is already occupied by data.

**owner.** The user who has the highest level of access authority to a data object or action, as defined by the object or action.

**pad.** To fill unused positions in a field with dummy data, usually zeros or blanks.

**page.** A block of instructions, data, or both.

**page space minidisk.** The area on a fixed disk that temporarily stores instructions or data currently being run. See also *minidisk*.

**pagination.** The process of adjusting text to fit within margins and/or page boundaries.

**paging.** The action of transferring instructions, data, or both between real storage and external page storage.

---

**parallel processing.** The condition in which multiple tasks are being performed simultaneously within the same activity.

**parameter.** Information that the user supplies to a panel, command, or function.

**parent.** Pertaining to a secured resource, either a file or library, whose user list is shared with one or more other files or libraries. Contrast with *child*.

**parent directory.** The directory one level above the current directory.

**partition.** See *minidisk*.

**password.** A string of characters that, when entered along with a user identification, allows an operator to sign on to the system.

**password security.** A program product option that helps prevent the unauthorized use of a display station, by checking the password entered by each operator at sign-on.

**path name.** See *full path name* and *relative path name*.

**pattern-matching character.** Special characters such as \* or ? that can be used in search patterns. Some used in a file specification to match one or more characters. For example, placing a ? in a file specification means any character can be in that position. Pattern-matching characters are also called wildcards.

**permission code.** A three-digit octal code, or a nine-letter alphabetic code, indicating the access permissions. The access permissions are read, write, and execute.

**permission field.** One of the three-character fields within the permissions column of a directory listing indicating the read, write, and run permissions for the file or directory owner, group, and all others.

**phase.** One of several stages file system checking and repair performed by the **fsck** command.

**physical device.** See *device*.

**physical file.** An indexed file containing data for which one or more alternative indexes have been created.

**physical record.** (1) A group of records recorded or processed as a unit. Same as *block*. (2) A unit of data moved into or out of the computer.

**PID.** See *process ID*.

**pipe.** To direct the data so that the output from one process becomes the input to another process.

**pipeline.** A direct, one-way connection between two or more processes.

**pitch.** A unit of width of typewriter type, based on the number of times a letter can be set in a linear inch. For example, 10-pitch type has 10 characters per inch.

**platen.** The support mechanism for paper on a printer, commonly cylindrical, against which printing mechanisms strike to produce an impression.

**pointer.** A logical connection between physical blocks.

**port.** (1) To make the programming changes necessary to allow a program that runs on one type of computer to run on another type of computer. (2) An access point for data input to or data output from a computer system. See *connector*.

**position.** The location of a character in a series, as in a record, a displayed message, or a computer printout.

**positional parameter.** A shell facility for assigning values from the command line to variables in a program.

---

**print queue.** A file containing a list of the names of files waiting to be printed.

**printout.** Information from the computer produced by a printer.

**priority.** The relative ranking of items. For example, a job with high priority in the job queue will be run before one with medium or low priority.

**priority number.** A number that establishes the relative priority of printer requests.

**privileged user.** The account with superuser authority.

**problem determination.** The process of identifying why the system is not working. Often this process identifies programs, equipment, data communications facilities, or user errors as the source of the problem.

**problem determination procedure.** A prescribed sequence of steps aimed at recovery from, or circumvention of, problem conditions.

**procedure.** See *shell procedure*.

**process.** (1) A sequence of actions required to produce a desired result. (2) An entity receiving a portion of the processor's time for executing a program. (3) An activity within the system begun by entering a command, running a shell program, or being started by another process.

**process accounting.** An analysis of the use each process makes of the processing unit, memory, and I/O resources.

**process ID (PID).** A unique number assigned to a process that is running.

**profile.** (1) A file containing customized settings for a system or user (2) Data describing the significant features of a user, program, or device.

**program.** A file containing a set of instructions conforming to a particular programming language syntax.

**prompt.** A displayed request for information or operator action.

**propagation time.** The time necessary for a signal to travel from one point on a communications line to another.

**qdaemon.** The daemon process that maintains a list of outstanding jobs and sends them to the specified device at the appropriate time.

**queue.** A line or list formed by items waiting to be processed.

**queued message.** A message from the system that is added to a list of messages stored in a file for viewing by the user at a later time. This is in contrast to a message that is sent directly to the screen for the user to see immediately.

**quit.** A key, command, or action that tells the system to return to a previous state or stop a process.

**quote.** To mask the special meaning of certain characters; to cause them to be taken literally.

**random access.** An access mode in which records can be read from, written to, or removed from a file in any order.

**readonly.** Pertaining to file system mounting, a condition that allows data to be read, but not modified.

**recovery procedure.** (1) An action performed by the operator when an error message appears on the display screen. Usually, this action permits the program to continue or permits the operator to run the next job. (2) The method of returning the system to the point where a major system error occurred and running the recent critical jobs again.

**redirect.** To divert data from a process to a file or device to which it would not normally go.

---

**reference count.** In an i-node, a record of the total number of directory entries that refer to the i-node.

**relational expression.** A logical statement describing the relationship (such as greater than or equal) of two arithmetic expressions or data items.

**relational operator.** The reserved words or symbols used to express a relational condition or a relational expression.

**relative address.** An address specified relative to the address of a symbol. When a program is relocated, the addresses themselves will change, but the specification of relative addresses remains the same.

**relative addressing.** A means of addressing instructions and data areas by designating their locations relative to some symbol.

**relative path name.** The name of a directory or file expressed as a sequence of directories followed by a file name, beginning from the current directory.

**remote.** Pertaining to a system or device that is connected to your system through a communications line. Contrast with *local*.

**reserved character.** A character or symbol that has a special (non-literal) meaning unless quoted.

**reserved word.** A word that is defined in a programming language for a special purpose, and that must not appear as a user-declared identifier.

**reset.** To return a device or circuit to a clear state.

**restore.** To return to an original value or image. For example, to restore a library from diskette.

**right adjust.** The process of aligning lines of text at the right margin or tab setting such that

the rightmost character in the line or file is in the rightmost position.

**right justify.** See right align.

**right margin.** The area on a page between the last text character and the right upper edge.

**right-adjust.** To place or move an entry in a field so that the rightmost character of the field is in the rightmost position. Contrast with *left-adjust*.

**root.** Another name sometimes used for superuser.

**root directory.** The top level of a tree-structured directory system.

**root file system.** The basic AIX Operating System file system, which contains operating system files and onto which other file systems can be mounted. The root file system is the file system that contains the files that are run to start the system running.

**routine.** A set of statements in a program causing the system to perform an operation or a series of related operations.

**run.** To cause a program, utility, or other machine function to be performed.

**run-time environment.** A collection of subroutines and shell variables that provide commonly used functions and information for system components.

**scratch file.** A file, usually used as a work file, that exists until the program that uses it ends.

**screen.** See *display screen*.

**scroll.** To move information vertically or horizontally to bring into view information that is outside the display screen boundaries.

**sector.** (1) An area on a disk track or a diskette track reserved to record information. (2) The smallest amount of information that

---

can be written to or read from a disk or diskette during a single read or write operation.

**security.** The protection of data, system operations, and devices from accidental or intentional ruin, damage, or exposure.

**segment.** A contiguous area of virtual storage allocated to a job or system task. A program segment can be run by itself, even if the whole program is not in main storage.

**separator.** A character used to separate parts of a command or file.

**sequential access.** An access method in which records are read from, written to, or removed from a file based on the logical order of the records in the file.

**session records.** In the accounting system, a record of time connected and line usage for connected display stations, produced from log in and log out records.

**set flags.** Flags that can be put into effect with the shell set command.

**shared printer.** A printer that is used by more than one work station.

**shell.** See *shell program*.

**shell procedure.** A series of commands combined in a file that carry out a particular function when the file is run or when the file is specified as an argument to the sh command. Shell procedures are frequently called shell scripts.

**shell program.** A program that accepts and interprets commands for the operating system (there is an AIX shell program and a DOS shell program).

**shell prompt.** The character string on the command line indicating the the system can accept a command (typically the \$ character).

**shell script.** See *shell procedure*.

**shell variables.** Facilities of the shell program for assigning variable values to constant names.

**size field.** In an i-node, a field that indicates the size, in bytes, of the file associated with the i-node.

**software.** Programs.

**sort.** To rearrange some or all of a group of items based upon the contents or characteristics of those items.

**source diskette.** The diskette containing data to be copied, compared, restored, or backed up.

**source program.** A set of instructions written in a programming language, that must be translated to machine language compiled before the program can be run.

**special character.** A character other than an alphabetic or numeric character. For example; \*, +, and % are special characters.

**special file.** Special files are used in the AIX system to provide an interface to input/output devices. There is at least one special file for each device connected to the computer. Contrast with *directory* and *file*. See also *block special file* and *character special file*.

**spool files.** Files used in the transmission of data among devices.

**standalone shell.** A limited version of the shell program used for system maintenance.

**standalone work station.** A work station that can be used to preform tasks independent of (without being connected to) other resources such as servers or host systems.

**standard error.** The place where many programs place error messages.

**standard input.** The primary source of data going into a command. Standard input comes from the keyboard unless redirection or piping is used, in which case standard input can be



---

from a file or the output from another command.

**standard output.** The primary destination of data coming from a command. Standard output goes to the display unless redirection or piping is used, in which case standard output can be to a file or another command.

**stanza.** A group of lines in a file that together have a common function. Stanzas are usually separated by blank lines, and each stanza has a name.

**statement.** An instruction in a program or procedure.

**status.** (1) The current condition or state of a program or device. For example, the status of a printer. (2) The condition of the hardware or software, usually represented in a status code.

**storage.** (1) The location of saved information. (2) In contrast to memory, the saving of information on physical devices such as disk or tape. See *memory*.

**storage device.** A device for storing and/or retrieving data.

**string.** A linear sequence of entities such as characters or physical elements. Examples of strings are alphabetic string, binary element string, bit string, character string, search string, and symbol string.

**su.** See *superuser*.

**subdirectory.** A directory contained within another directory in the file system hierarchy.

**subprogram.** A program invoked by another program, such as a subshell.

**subroutine.** (1) A sequenced set of statements that may be used in one or more computer programs and at one or more points in a computer program. (2) A routine that can be part of another routine.

**subscript.** An integer or variable whose value refers to a particular element in a table or an array.

**subshell.** An instance of the shell program started from an existing shell program.

**substring.** A part of a character string.

**subsystem.** A secondary or subordinate system, usually capable of operating independently of, or synchronously with, a controlling system.

**superblock.** The most critical part of the file system containing information about every allocation or deallocation of a block in the file system.

**superuser (su).** The user who can operate without the restrictions designed to prevent data loss or damage to the system (user ID 0).

**superuser authority.** The unrestricted ability to access and modify any part of the operating system that is associated with the user who manages the system. The authority obtained when one logs in as *root*.

**system.** The computer and its associated devices and programs.

**system call.** A request by an active process for a service by the system kernel.

**system customization.** A process of specifying the devices, programs, and users for a particular data processing system.

**system date.** The date assigned by the system user during setup and maintained by the system.

**system dump.** A copy of memory made whenever an error stops the system. Contrast with *task dump*.

**system management.** The tasks involved in maintaining the system in good working order and modifying the system to meet changing requirements.

---

**system parameters.** See *kernel parameters*.

**system profile.** A file containing the default values used in system operations.

**system unit.** The part of the system that contains the processing unit, the disk drives, and the diskette drives.

**system user.** A person who uses a computer system.

**target diskette.** The diskette to be used to receive data from a source diskette.

**task.** A basic unit of work to be performed. Examples are a user task, a server task, and a processor task.

**task dump.** A copy of memory associated with a program that failed (and its data). Contrast with *system dump*.

**terminal.** An input/output device containing a keyboard and either a display device or a printer. Terminals usually are connected to a computer and allow a person to interact with the computer.

**text.** A type of data consisting of a set of linguistic characters (for example, alphabet, numbers, and symbols) and formatting controls.

**text application.** A program defined for the purpose of processing text data (for example, memos, reports, and letters).

**text editing program.** See *editor* and *text application*.

**texttab.** A kernel parameter establishing the size of the text table, in memory, that contains one entry each active shared program text segment.

**trace.** To record data that provides a history of events occurring in the system.

**trace table.** A storage area into which a record of the performance of computer program instructions is stored.

**track.** A circular path on the surface of a fixed disk or diskette on which information is magnetically recorded and from which recorded information is read.

**trap.** An unprogrammed, hardware-initiated jump to a specific address. Occurs as a result of an error or certain other conditions.

**tree-structured directories.** A method for connecting directories such that each directory is listed in another directory except for the root directory, which is at the top of the tree.

**truncate.** To shorten a field or statement to a specified length.

**typematic key.** A key that repeats its function multiple times when held down.

**typestyle.** Characters of a given size, style and design.

**Uid.** See *user number*.

**update.** An improvement for some part of the system.

**user.** The name associated with an account.

**user account.** See *account*.

**user ID.** See *user number*.

**user name.** A name that uniquely identifies a user to the system.

**user number (Uid).** (1) A unique number identifying an operator to the system. This string of characters limits the functions and information the operator is allowed to use. The Uid can often be substituted in commands that take a user's name as an argument.

**user profile.** A file containing a description of user characteristics and defaults (for example, printer assignment, formats, group ID) to be conveyed to the system while the user is signed on.

**utility.** A service; in programming, a program that performs a common service function.

---

**valid.** (1) Allowed. (2) True, in conforming to an appropriate standard or authority.

**value.** (1) In Usability Services, information selected or typed into a pop-up. (2) A set of characters or a quantity associated with a parameter or name. (3) In programming, the contents of a storage location.

**variable.** A name used to represent a data item whose value can change while the program is running. Contrast with *constant*.

**verify.** To confirm the correctness of something.

**version.** Information in addition to an object's name that identifies different modification levels of the same logical object.

**virtual device.** A device that appears to the user as a separate entity but is actually a shared portion of a real device. For example, several virtual terminals may exist simultaneously, but only one is active at any given time.

**virtual machine.** A functional simulation of a computer and its related devices.

**virtual machine interface (VMI).** A software interface between work stations and the operating system. The VMI shields operating system software from hardware changes and low-level interfaces and provides for concurrent execution of multiple virtual machines.

**virtual resource manager (VRM).** A set of programs that manage the hardware resources (main storage, disk storage, display stations, and printers) of the system so that these

resources can be used independently of each other.

**virtual resources.** See *virtual resource manager*.

**virtual storage.** Addressable space that appears to be real storage. From virtual storage, instructions and data are mapped into real storage locations.

**virtual terminal.** Any of several logical equivalents of a display station available at a single physical display station.

**Volume ID (Vol ID).** A series of characters recorded on the diskette used to identify the diskette to the user and to the system.

**VRM.** See *virtual resource manager*.

**wildcard.** See *pattern-matching characters*.

**word.** A contiguous series of 32 bits (4 bytes) in storage, addressable as a unit. The address of the first byte of a word is evenly divisible by four.

**work file.** A file used for temporary storage of data being processed.

**work station.** A device at which an individual may transmit information to, or receive information from, a computer for the purpose of performing a task, for example, a display station or printer. See *programmable work station* and *dependent work station*.

**working directory.** See *current directory*.

**wrap around.** Movement of the point of reference in a file from the end of one line to the beginning of the next, or from one end of a file to the other.

## Special Characters

\$- 647  
 \$! 647  
 \$\$ 647  
 \$? 647  
 \$# 647

## A

abs command 693  
 access times of a file, changing 760  
 accounting  
   ASCII format 47  
   ASCII summary format 37  
   billing summary file 606  
   binary summary format 37  
   combining total accounting files 46  
   connect 42  
   daily 606  
   disk 32  
   line-usage summary 42  
   login 33  
   merging total accounting files 46  
   monthly reports 33  
   process 32, 48  
   reports 33  
   session 33  
   shell procedures 31  
   start 34  
   turn off process 34  
   usage summaries 36  
 accounting commands  
   acctcms 36  
   acctcom 38  
   acctcon1 42  
   acctcon2 43

acctdisk 44  
 acctdusg 44  
 acctmerg 46  
 accton 49  
 acctprc1 48  
 acctprc2 49  
 acctwtmp 345  
 chargefee 32  
 ckpacct 32  
 dodisk 32  
 fwtmp 345  
 lastlogin 33  
 monacct 33  
 nulladm 33  
 prctmp 33  
 prdaily 33  
 prtacct 34  
 runacct 606  
 shutacct 34  
 startup 34  
 turnacct 34  
 wttmpfix 346  
 accounting file 33  
 accounting files  
   /usr/adm/acct/fiscal 33  
   /usr/adm/acct/nite/active 606  
   /usr/adm/acct/nite/ctmp 33  
   /usr/adm/acct/nite/lastdate 606  
   /usr/adm/acct/nite/lock 606  
   /usr/adm/acct/nite/lock1 606  
   /usr/adm/acct/nite/statefile 606  
   /usr/adm/acct/sum 33  
   /usr/adm/acct/sum/loginlog 33  
   /usr/adm/acct/sum/rprt 33  
   /usr/adm/fee 32  
   /usr/adm/pacct 32  
   acctcom 38  
   accton 49  
   ckpacct 32  
   turnacct 34

---

- /usr/adm/wtmp
  - acctcon1 42
  - billing summary file 606
  - creating 33
- accounting records
  - ASCII 48
  - ASCII format 42
  - converting ASCII to binary 345
  - converting binary to ASCII 345
  - display 38
  - examining connect records 345
  - login session 42
  - repairing wtmp records 346
  - session 42
  - total accounting login session 43
- accounting report, process 38
- accounting, disk usage 44
- acctcms command 36-37
- acctcom command 38-41
- acctcon1 command 42-43
- acctcon2 command 43
- acctdisk command 44
- acctdusg command 44
- acctmerg command 46-47
- accton command 49
- acctprc2 command 49
- acctwtmp command 345
- activity graph, system 612
- activity manager 50
- activity reporter, system 610, 614
- actman command 50
- adb command 50.1-50.8
- adding
  - devices 241
  - groups 802
  - header flags, SCCS 55
  - users 802
  - users, SCCS 55
- adduser command 802-804
- admin command 51-57
- Advanced Floating-Point Accelerator 115

- Advanced Processor Card 115
- af command 694
- ar command 58-61
- arbitrary precision arithmetic 83
- arithmetic game 62-63
- arithmetic, shell variable 317
- as command 64-65, 115
- assembler 64
- assembling
  - source code
    - as 64
    - asm 64
    - cc 113
    - masm 64
- at command 66-69
- awk command 70-74

## B

- back game 75
- backing up files 76
- backup command 76-79
- banner command 80
- bar command 709
- basename command 81-82
- batch command 66-69
- bc command 83-87
- bdiff command 88-89
- bel command 387
- belonging to different groups 385
- bfs command 90-93
- billing summary file, accounting 606
- bj game 94
- blackjack game 94
- block count of a file, display 726
- branching from nonleaf deltas 363
- break command 654
- bs command 95-105
- bucket command 703

---

  
**C****C Language programming**

See also managing programs

See also programming

assembling source code 113

commands

ar 58

as 64

cb 111

cc 112

cflow 125

cpp 163

fcc 113

lint 446

vcc 113

vrmfmt 113

cross-reference listing 217

files

a.out 113

formatting source code 111

linking object files 113

maintaining linkage libraries 58

preprocessing source code 113

syntax checking 446

cal command 106

calculating

CPU factor 40

CPU time 39

hog factor 39

calculator program 83

calculator, desk 222

calendar command 107-108

calprog program 108

case command 653

cat command 109-110

cb command 111

cc command 112-120.1, 135

cd command 121-122, 654

cdc command 123-124

CDPATH 645

ceil command 695

cflow command 125-126

changing

ASCII accounting records to binary 345

binary accounting records to ASCII 345

changing permission codes 128

current directory 121

devices 241

files, SCCS 236

format of a file 507

group identification 510

group ownership 126.1

groups 802

login environment 510

owner-ID of files or directories 132

password 546

primary group 510

root directory 134

SCCS delta comments 123

system parameters 133

users 802

changing Distributed Services ipc queues  
table 414.1changing Distributed Services network  
user/groups table 784

changing Distributed Services node table 506.1

changing state values 272.4

chargefee command 32

charting

external references 125

checkcw command 215-216

checkeq command 300-301

checking process accounting files 32

checkmm command 492

checksum of a file, display 726

chgrp command 126.1-127

changing

group ownership 126.1

chmod command 128-131

chown command 132-133

chparm command 133

chroot command 134-135

ckpacct command 32

ckprereq command 406

clearing an i-node 136

clri command 136-137

cmp command 138-139

col command 140-141

collating sequence

collating sequence

---

- csch command 183
  - ctab command 204
  - sh command 640
  - sort command 672
- csch command 183
- ctab command 204
- equivalence classes
  - ctab command 204
- li command 438
- ls command 462
- NLCTAB environment variable 645
- sh command 640
- sort command 672
- colors
  - setting active display palette 258
  - setting background display 258
  - setting foreground display 258
- comb command 142-143
- combining
  - deltas, SCCS 142
  - total accounting files 46
- comm command 144-145
- command execution environment 298
- command line flag parsing 367
- command usage summary 36
- commands
  - See accounting commands
  - See C Language programming
  - See communication commands
  - See editors
  - See filter commands
  - See graphics commands
  - See maintenance commands
  - See Multi-User Services commands
  - See programming
  - See reading standard input
  - See SCCS, commands
  - See system group commands
  - See text processing commands
  - See writing to standard output
- communication commands
  - confer 146
  - connect 152
  - mesg 484
  - news 512
  - uuclean 805
  - uucp 807
  - uulog 807
  - uname 807
  - uupick 815
  - uustat 810
  - uusub command 813
  - uuto 815
  - uux 818
  - wall 845
  - who 850
  - 300 863
  - 4014 865
  - 450 866
- communication, inter-process status 411
- comparing
  - directories
    - dircmp 254
  - files 627
    - bdiff 88
    - cmp 138
    - diff 246
    - diffmk 252
    - diff3 249
    - dircmp 254
  - SCCS files 618
- compilers
  - bs 95
  - cc 112
  - sno 670
- compress program 683
- compressing files 543
- concatenate files 109
- concurrent groups 385
- conditional expressions, evaluating 750
- confer command 146-149
- config command 150-151
- configuration information 150
- connect accounting 42
- connect command 152-155
- consistency check and repair of files
  - dfsck command 335
  - fsck command 333
- constant-width text 213
- constructing a file system 487
- contents of directory, listing 267, 437
- context split 202

---

continue command 654  
converting  
  ASCII accounting records to binary 345  
  binary accounting records to ASCII 345  
copy command 156-157  
copying  
  AIX files  
    copy 156  
    cp 156  
  DOS files  
    dosread 269  
    doswrite 271  
cor command 704  
cp command 156-157  
cpio command 158-162  
cpp command 115, 163-166  
CPU factor computation 40  
CPU time computation 39  
craps game 167  
crash command 168-171  
creating  
  C program cross-reference listing 217  
  delta, SCCS 236  
  mount table 635  
  SCCS files 51  
  special file 490  
  specified version of an SCCS file 359  
cron command 172-173, 606  
  used with the sa1 command 611  
  used with the sa2 command 611  
crontab command 174-176  
cross-reference listing, C program 217  
csh command 177-201  
csplit command 202-203  
ctab command 204-207  
ctags command 208-209  
current directory  
current directory, changing 121  
csum command 696  
cut command 210-211  
cvid command 212  
cvrtopt command 387  
cw command 213-216  
  used in pipeline with nroff 526  
cxref command 217-218

## D

daily accounting 606  
database operator 417  
date command 219-221, 272.3, 272.6  
dc command 222-225  
dcopy command 226-227  
dd command 228-231  
debugger, file system 338  
defining shell functions 654  
defkey command 232-233  
del command 234-235  
deleting  
  delta from SCCS file 604  
  devices 241  
  directories  
    rm 601  
    rmdir 605  
  DOS files 266  
  files  
    del 234  
    rm 601  
    skulker 667  
  groups 802  
  repeated words 792  
  users 802  
  users, SCCS 55  
delta command 236-238  
delta summary of SCCS file 365  
deltas, branching from nonleaf 363  
demon, error-logging 303  
demon, error-logging termination 309  
deroff command 239-240  
description file, make command 479  
desk calculator 222  
device (special) files  
  /dev/null  
    acctcom 38  
    standard input assigned to 38  
  adding 241  
  changing 241  
  creating 490  
  deleting 241



---

device name 242  
 devices  
 devices command 241  
 devnm command 242-243  
 df command 244  
 dfsck command 335-337  
 di command 437-443  
 diff command 138, 246-248  
 diffmk command 252-253  
 diff3 command 249-251  
 dircmp command 254-255  
 directories  
   changing  
     owner-ID 132  
   comparing  
     dircmp 254  
   listing contents  
     di 437  
     DOS directories 267  
     li 437  
     ls 461  
   removing  
     rm 601  
     rmdir 605  
 directory 504  
   change root 134  
   changing  
   changing current 121  
   create 486  
   moving 504  
   renaming 504  
   return path name 81  
 directory contents, listing 267, 437  
 dirname command 81-82  
 disk usage accounting 44  
 disk usage summary 273  
 diskusg command 256-257  
 display command 258-261  
 display station  
   changing DMA pinned page 259  
   setting active color palette 258  
   setting background colors 258  
   setting fonts 258  
   setting foreground colors 258  
 displaying  
   a calendar 106  
   accounting report 33  
   compressed files 543  
   connect accounting records 345  
   contents of i-nodes 415  
   corresponding group names and IDs 395  
   corresponding user names and IDs 395  
   current directory 589  
   date 219  
   documents formatted with the Memorandum  
     Macros 492  
   file checksum 726  
   files 90, 109  
   formatted files 553, 561  
   login name 456  
   news items 512  
   packed files 543  
   process accounting records 38  
   process status 579  
   profile data 571  
   SCCS file editing activity 609  
   session record 33  
   squeezed files 543  
   system images 168  
   system parameters 133  
   total accounting report 34  
 Distributed Services  
   dsldxprof 272.2  
 dividing a file into pieces 686  
 DMA channel, setting 634.1  
 dodisk command 32  
 dos command 262-265  
 dosdel command 266  
 dosdir command 267-268  
 dosread command 269-270  
 doswrite 271-272  
 drill in arithmetic skills 62  
 dsipc command 272.1  
 dsldxprof command 272.2  
 dsstate command 272.4-272.5  
 dsxlate command 272.6  
 dtoc command 757  
 du command 273-274  
 dump command 275-276  
 dump, extracting error records 302  
 dump, octal 538  
 dumpfmt command 277

---

## E

echo command 278-279  
ed command 280-291  
edit command 292-297  
editors  
  ed 280  
  edit 292  
  ex 312  
  ged 350  
  red 280  
  sed 629  
  vedit 832  
  vi 832  
  view 832  
egrep command 381-384  
end a process 422  
env command 298-299  
environment, changing login 510  
eqn command 300, 301  
  constructs removed by the deroff  
  command 239  
  used in pipeline with nroff 526  
  used with tbl 739  
erase command 348  
errdead command 302  
errdemon command 303-304  
error-logging demon 303  
error-logging demon termination 309  
error records extraction from dump 302  
error report 305  
errpd command 304, 308  
errpt command 305-308  
errstop command 309  
errupdate 311  
errupdate command 310  
eval command 654  
evaluating expressions  
  expr 317  
  test 750  
ex command 312-316  
examining  
  connect accounting records 345  
  contents of i-nodes 415  
  files 90

  system images 168  
  system parameters 133  
exec command 655  
exercising link system call 444  
exit command 655  
exp command 696  
expanding packed files 543  
export command 655  
expr command 317-320  
expression evaluation 317  
extended character support  
  See international character support  
external references, flow graph 125  
extract error records from dump 302

## F

factor command 321  
factoring a number 321  
false command 777  
fcc command 113  
ff command 322-322.2  
fgrep command 381-384  
file  
  display checksum 726  
file command 324-325  
file formats  
  acct 36, 48  
  ar 58  
  backup 76  
  tacct 46  
  utmp 345  
  wtmp 346  
file pattern search 381  
file system  
  See also device (special) files  
  See also maintenance commands  
  See also system files  
  backing up 76  
  make available for use 498  
  make unavailable for use 786  
  making 487  
  moving a directory 504  
  renaming a directory 504

---

- unmount 786
- file system debugger 338
- files
  - See also accounting files
  - See also device (special) files
  - See also SCCS, files
  - See also system files
  - a.out 113
  - backing up 76
  - calendar 107
  - changing
    - owner-ID 132
  - checking consistency
    - dfsck command 335
    - fsck command 333
  - comparing 627
    - cmp 138
    - diff 246
    - diffmk 252
    - diff3 249
    - dircmp 254
  - comparing large files 88
  - compressing 543
  - concatenating 109
  - copying
    - AIX files 156
    - DOS files 269, 271
  - creating SCCS files 51
  - deleting
    - del 234
    - DOS files 266
  - determining type 324
  - displaying 109
  - displaying formatted files 553, 561
  - expanding 543
  - finding 326
  - identifying the processes using a file 343
  - initializing SCCS files 51
  - linking 450
  - merge lines 547
  - merging 672
  - modifying the user mask 784.1
  - naming SCCS files 53
  - packing 543
  - parallel merging 547
  - removing
    - rm 601
    - skulker 667
  - repairing
    - dfsck command 335
    - fsck command 333
  - repairing damage 168
  - return base name 81
  - scanning 90
  - searching 90
  - searching for a pattern 381
  - serial merging 547
  - setting file-creation permission code
    - mask 784.1
  - sorting 672
  - squeezing 543
  - text
    - changing the format 507
  - transforming 228
  - translating 228
  - unpacking 543
  - unsqueezing 543
  - writing the last part 732
  - 3-way comparison 249
- filter commands
  - acctcom 38
  - acctcon1 42
  - acctmerg 46
  - awk 70
  - bdiff 88
  - cb 111
  - cmp 138
  - col 140
  - comb 142
  - cw 213
  - definition of 638
  - fwtmp 345
  - hp 392
  - nl 517
  - nroff 526
  - paste 547
  - ptx 584
  - tbl 739
  - troff 526
  - wtmpfix 346
- find command 326, 329
  - acctdusg 44

---

find hyphenated words 394  
find necessary order of files in an object  
  library 457  
fish game 330  
fixed minidisk information 485  
Floating-Point Accelerator 115  
floating point configuration 332.1  
floor command 696  
flow graph of external references 125  
fonts  
  setting virtual terminal 258  
for command 653  
format command 331  
formats  
  See file formats  
formatting C Language source code 111  
formatting text  
  constant-width text 213  
  for a phototypesetter 525  
  for a printing device 525  
  inverse linefeeds and half-linefeeds 140  
  mathematical text 300  
  tables for **nroff** 739  
  tables for **troff** 739  
fortune game 332  
forwarding mail 471  
fptype command 332.1  
free disk space, reporting 244  
fsck command 333-337  
fsdb command 136, 338-342  
fuser command 343-344  
fwtmp command 345

## G

games  
  arithmetic 62  
  back 75  
  backgammon 75  
  bj 94  
  blackjack 94  
  craps 167  
  fish 330  
  fortune 332

hangman 390  
moo 497  
number 537  
quiz 591  
ttt 780  
wump 856  
gamma command 697  
gd command 388  
gdev commands 347-349  
ged command 350-356  
gend command 357-358  
generating C program cross-reference  
  listing 217  
generating names from i-numbers 505  
get command 236, 237, 359-366  
getopt command 367-369  
gettext command 370-371  
getty command 372-374  
going to maintenance mode 663  
graph command 375-376  
graph, system activity 612  
graphical editor 350  
graphics command 377-378  
graphics commands  
  abs 693  
  af 694  
  bar 709  
  bel 387  
  bucket 703  
  ceil 695  
  cor 704  
  cusum 696  
  cvrtopt 387  
  dtoc 757  
  erase 348  
  exp 696  
  floor 696  
  gamma 697  
  gd 388  
  ged 350  
  gend 357  
  graph 375  
  graphics 377  
  gtop 388  
  hardcopy 348  
  hilo 705

---

hist 710  
hpd 347  
label 711  
list 697  
log 698  
lreg 705  
mean 706  
mod 699  
pair 699  
pd 388  
pie 712  
plot 713  
point 707  
power 700  
prime 692  
prod 707  
ptog 388  
qsort 708  
quit 388  
rand 692  
rank 708  
remcom 388  
root 700  
round 701  
siline 701  
sin 702  
spline 684  
subset 702  
td 348  
tekset 348  
title 715  
total 708  
tplot 762  
ttoc 758  
utility commands 386  
var 709  
vtoc 758  
whatis 388  
graphing  
  external references 125  
greek command 379-380  
grep command 381-384  
group  
  adding 802  
  changing 802  
  deleting 802

group identification, changing 510  
group IDs and names, displaying 395  
group membership 385  
group membership, display 385  
groups command 385  
grpck command 588  
gtop command 388  
guess a word 390

## H

hangman game 390  
hardcopy command 348  
hash command 655  
hashcheck command 682  
hashmake command 682  
header flags, SCCS 53, 55  
help command 391  
hilo command 705  
hist command 710  
hog factor computation 39  
HOME 645  
hp command 392-393  
hpd command 347-348  
hyphen command 394  
hyphenated words 394

## I

i-node content, displaying 415  
i-node examination 415  
i-numbers  
  generating names 505  
I/O counts 40  
id command 395  
ID, special user  
  adm 606  
  root 422, 606, 724, 725  
if command 653  
IFS 646  
init command 396-398

---

initialization, normal startup 594  
initializing  
    SCCS files 51  
install command 399-401  
installp command 402-408  
inter-process communication status 411  
interactive processor 83  
international character support  
    at command 67  
    batch command 67  
    collating sequence  
    csh command 183  
    ctab command 204  
    dd command 229  
    dfsck command 335  
    equivalence classes  
    fsck command 335  
    fsdb command 340  
    li command 438  
    ls command 462  
    od command 539  
    print command 568  
    ps command 580  
    sh command 645  
    sort command 672  
    stty command 721  
interpolating a smooth curve 684  
interpreters  
    bc 83  
    bs 95  
    sno 670  
Interprocess Communication key mapping  
    installation 272.1  
inudocm command 799-800  
inuupd command 800  
ipc queues table  
    access 414.1  
iperm command 409-410  
ipcs command 411-414  
ipctable command 414.1  
istat command 415-416

## J

join command 417-420  
joinconf command 146  
joining database files 417  
joint editing of an SCCS file 364

## K

kcore minutes, definition 37  
keyboard command 421  
keyboard, redefine 232  
kill all nonancestral processes 425  
kill command 422-424  
killall command 425

## L

label command 711  
language support  
    See international character support  
lastlogin command 33  
ld command 115, 427-431  
lex command 432-436  
li command 437-443, 605  
library maintainer 58  
library search order 116  
line command 443  
line editor 280  
line numbering filter 517  
line printer backend 459  
link command 444, 445  
link library maintaining 58  
linkage editor 427  
linking  
    files 450  
    object files  
        cc 113  
        ld 427  
lint command 446-449

---

list command 697  
 listing  
   directory contents  
     di 437  
     li 437  
     ls 461  
   DOS directory contents 267  
   file names for a file system 322  
   statistics for a file system 322  
 ln command 450-451  
 locator command 452  
 log command 698  
 logged error report 305  
 login command 453-455  
 login environment, changing 510  
 login session records 42  
 login shell subshell under the login shell. 658  
 LOGNAME 645  
 logname command 456  
 looking at  
   connect accounting records 345  
   contents of i-nodes 415  
   files 90  
   system images 168  
   system parameters 133  
 lorder command 457-458  
 lp command 459-460  
 lreg command 705  
 ls command 461-465

**M**

m4 command 465, 469  
 macro processor 465  
 MAIL 645  
 mail command 470-473  
 MAILCHECK 645  
 MAILMSG 645  
 MAILPATH 645  
 maintaining groups of programs 474  
 maintaining linkage libraries 58  
 maintenance commands  
   backup 76  
   clri 136  
   cpio 158  
   mount 498  
   umount 786  
   unmount 786  
 maintenance mode, going to 663  
 make a directory 486  
 make a tags file 208  
 make command 474-480  
 makekey command 481  
 making a file system 487  
 making two files the same 88  
 manager, virtual terminals 50  
 managing programs  
   See also programming  
   See also SCCS  
   make 474  
 mant command 495  
 marking differences between files 252  
 mdrc command 482-483  
 mean command 706  
 membership, display group 385  
 merging files 672  
 merging lines in files 547  
 merging total accounting files 46  
 mesg command 484  
 message queue removal 409  
 messages, permitting 484  
 messages, refusing 484  
 messages, send 470  
 messages, sending 853  
 minidisks 485  
 minidisks command 485  
 mkdir command 486  
 mkfs command 487-489  
 mknod command 490-491  
 mm command 492-494  
 mmt command 495-496  
 mod command 699  
 modification request number 237  
 modification times of a file, changing 760  
 modifying  
   changing permission codes 128  
   current directory 121  
   devices 241  
   files, SCCS 236  
   group identification 510

---

group ownership 126.1  
groups 802  
login environment 510  
owner-ID of files or directories 132  
password 546  
primary group 510  
root directory 134  
SCCS delta comments 123  
system parameters 133  
users 802  
modifying access times of a file 760  
modifying modification times of a file 760  
monacct command 33  
moo game 497  
mount a file system 498  
mount command 498-501  
mount table, creating 635  
move command 502-503  
moving a directory 504  
moving files 502  
mt command 495  
Multi-User Services commands  
  acctcms 36  
  acctcom 38  
  acctcon1 42  
  acctcon2 43  
  acctdisk 44  
  acctdusg 44  
  acctmerg 46  
  accton 49  
  acctprc1 48  
  acctprc2 49  
  acctwtmp 345  
  chargefee 32  
  ckpacct 32  
  dodisk 32  
  fwtmp 345  
  lastlogin 33  
  mesg 484  
  monacct 33  
  nulladm 33  
  prctmp 33  
  prdaily 33

prtacct 34  
runacct 606  
sar 614  
shutacct 34  
startup 34  
turnacct 34  
who 850  
wtmpfix 346  
300 863  
4014 865  
450 866  
mv command 502-503  
mmdir command 504  
mvmd command 407-408  
mvt command 495

## N

ncheck command 505-506  
ndtable command 506.1  
neqn command 300-301  
  used with tbl 739  
network user/groups table  
  access 784  
newform command 507-509  
newgrp command 510-511, 655  
news command 512-514  
nice command 515-516  
nl command 517-520  
NLCTAB 645  
nm command 521-522  
node table  
  access 506.1  
nohup command 523-524  
normal startup initialization 594  
nroff command 525-530  
  tbl, preprocessor 739  
nulladm command 33  
number factoring 321  
number game 537  
numbering lines 517



---

## O

object library  
  ordering relation 457  
octal dump 538  
od command 538-540  
open command 541, 542  
  used after actman command 50

## P

pack command 543-545  
pair command 699  
parallel merging of lines in files 547  
parameters  
  work station  
    erase 372  
    kill 372  
    logmodes 372  
    owner 373  
    parity 372  
    program 373  
    protection 373  
    runmodes 372, 373  
    special purpose options 373  
    speed 372  
parameters, setting work station 717  
  terminal mapping 721  
parsing command line flags 367  
passwd command 546  
password, change 546  
paste command 547-549  
PATH 646  
path name, return directory 81  
pattern matching  
  acctcom 40  
  awk 70  
pattern, search for 381  
pcat command 543-545  
pd command 388  
pdisable command 550-552  
penable command 550-552  
perform disk accounting functions 32  
perform monthly accounting 33  
performing process accounting 48  
permission codes, changing 128  
  changing  
    permissions 128  
permitting messages 484  
pg command 553-556  
phold command 550-552  
pie command 712  
piobe command 557-560  
pipe fitting 746  
pipeline  
  asynchronous execution 638  
  conditional execution 638  
  definition of 638  
  sequential execution 638  
plot command 713  
point command 707  
port characteristics 372  
port characteristics, setting 372  
power command 700  
pr command 561-563  
prctmp command 33  
prdaily command 33, 606  
precision arithmetic 83  
preprocessing  
  source code  
    cc 113  
    cpp 163  
preprocessor  
  macro 465  
primary group 385  
primary group, changing 510  
prime command 692  
print command 566-570  
printer backend 459  
printing  
  a calendar 106  
  accounting report 33  
  compressed files 543  
  corresponding group names and IDs 395  
  corresponding user names and IDs 395  
  current directory 589  
  date 219

---

documents formatted with the Memorandum  
  Macros 492  
file checksum 726  
formatted files 553, 561  
login name 456  
news items 512  
packed files 543  
process accounting records 38  
process status 579  
profile data 571  
SCCS file editing activity 609  
session record 33  
squeezed files 543  
total accounting report 34  
priority, running a command 515  
process a report of logged errors 305  
process accounting 48  
process accounting records, display 38  
process accounting report 38  
process accounting, turn off 34  
  /usr/adm/wtmp  
  shutacct 34  
process suspension 422  
prod command 707  
producing C program cross-reference  
  listing 217  
prof command 116, 571-572  
profiler commands 564  
profiling the operating system 564  
program checking, C programs 446  
program maintenance 474  
program update 800  
program updating 474  
programming  
  See also managing programs  
  assembler  
  as 64  
  assembling source code 113  
  awk command 70  
  bs 95  
  C Language  
  cb 111  
  cc 112  
  cflow 125  
  cpp 163  
  fcc 113  
  formatting source code 111  
  lint 446  
  vcc 113  
  vrmfmt 113  
  debugging programs  
  files  
  a.out 113  
  linking object files 113  
  managing programs  
  awk 70  
  make 474  
  messages  
  miscellaneous languages  
  ar 58  
  awk 70  
  preprocessing source code 113  
  the shell  
  proto command 573  
  prs command 574-578  
  prtacct command 34  
  ps command 579-583  
  PS1 646  
  PS2 646  
  ptecms.awk program 35  
  ptelus.awk program 35  
  ptog command 388  
  ptx command 584-585  
  puttext command 586-587  
  pwck command 588  
  pwd command 589, 655

Q

qdaemon command 590  
qsort command 708  
query terminal characteristics 748  
quit command 388  
quiz game 591-593

---

**R**

- rand command 692
- rank command 708
- rc command 594
  - startup 34
- read command 655
- read operations 40
- reading one line 443
- reading standard input
  - acctconl 42
  - acctmerg 46
  - as 64
  - at 66
  - awk 70
  - batch 66
  - bdiff 88
  - cb 111
  - cmp 138
  - comb 142
  - fwtmp 345
  - wtmpfix 346
- readonly command 655
- recovering from a system crash 168
- red command 280
- redefine keyboard 232
- refusing messages 484
- regcmp command 595
- rejecting lines common to two sorted files 144
- relational database operator 417
- remcom command 388
- reminder service 107
- remote system mail 471
- remote system, connection 152
- remotely-settable hardware tabs 729
- remove a message queue 409
- remove a semaphore set 409
- remove a shared memory id 409
- remove command 34
- removing
  - delta from SCCS file 604
  - devices 241
  - directories
    - rm 601
    - rmdir 605
- DOS files 266
- files
  - del 234
  - rm 601
  - skulker 667
- groups 802
- repeated words 792
- users 802
- users, SCCS 55
- renaming a directory 504
- renaming files 502
- repairing damaged files 168
- repairing wtmp records 346
- repeated words, deleting 792
- report process data and system activity 755
- report, process accounting 38
- reporting free disk space 244
- reports
  - SCCS 143
- reports, accounting
- restore command 596-600
- return base file name or directory path name 81
- return command 655
- returning a true or false value 777
- rm command 601-603, 605
- rmail command 471
- rmdel command 604
- rmdir command 605
- root command 700
- root directory, changing 134
- round command 701
- rsh command 658
- runacct command 606-608
- running a command at low priority 515
- running commands at a later time 66

**S**

- sact command 609
- sadc command 610-611
- sag command 612-613
- sar command 614-617
- saving files 76

---

sa1 command 610-611  
 sa2 command 610-611  
 scanning files 90  
 SCCS  
   branching from leaf deltas 363  
   changing delta comments 123  
   checking the structure of SCCS files 55  
   commands  
     admin 51  
     cdc 123  
     comb 142  
     delta 236  
     get 359  
     help 391  
     prs 574  
     sact 609  
     sccsdiff 618  
     unget 790  
     val 821  
     what 848  
   creating a file 51  
   creating a specified version of a file 359  
   Data Keywords 574  
   delta summary 365  
   files  
     auxiliary files 360  
     creating a delta 236  
     g-file 53, 236, 237, 359, 360, 365  
     l-file 360, 365  
     lock file 53  
     p-file 360, 361, 364  
     s-file 360  
     x-file 53  
     z-file 53, 360  
   getting help information 391  
   header flags 53  
   identification keywords 362  
   initializing a file 91  
   interpreting errors 391  
   joint editing of files 364  
   modification request (MR) number 56, 237  
   Modification Requests 123  
   naming a file 53  
   recalculating the SCCS file checksum 56  
   removing a delta 604  
   reports 143  
     SID (SCCS Identification) 237  
       specifying version date cutoff 364  
       z-file 361  
   sccsdiff command 618  
   scheduling commands 66  
   scheduling queue requests 590  
   sdb command 619-626  
   sdiff command 627-628  
   search order, library 116  
   searching files 90  
   sed command 629-634  
   segment files 202  
   selecting fields from a file 210  
   selecting lines common to two sorted files 144  
   semaphore set removal 409  
   sending messages 470, 853  
   serial merging of lines in files 547  
   session records 42  
   set command 367, 656  
   set environment 298  
   setdma command 634.1  
   setmnt command 635-636  
   setting DMA channel 634.1  
   setting file-creation permission code  
     mask 784.1  
   setting port characteristics 372  
   setting tabs on a work station 729  
   setting the date 219  
   setting the parameters for a work station 717  
   sh command 637-659  
     See also shell  
       command line separators and  
       terminators 638  
   SHACCT 646  
   shared memory id removal 409  
   shell 646  
     actman 50  
     blank interpretation 655  
     built-in commands 654  
     command environment 642  
     command execution 638  
     command-line substitution 641  
     command substitution 647  
     conditional substitution 643  
     control commands 653  
     diagnostic output 650

---

file descriptors 649  
file-name substitution 640  
inline input documents 650  
positional parameters 641  
predefined special variables  
profile file 639  
quoting mechanisms 648  
redirection of input and output 649, 653  
running the shell 658  
shell variables  
  \$. 647  
  \$! 647  
  \$\$ 647  
  \$? 647  
  \$# 647  
  CDPATH 645  
  HOME 645  
  IFS 646  
  LOGNAME 645  
  MAIL 645  
  MAILCHECK 645  
  MAILMSG 645  
  MAILPATH 645  
  NLCTAB 645  
  PATH 646  
  PS1 646  
  PS2 646  
  SHACCT 646  
  SHELL 646  
  TIMEOUT 647  
signals 639  
standard input and output 649  
subshell 639  
summary of redirection options 652  
user-defined variables 641  
variables 641  
shell environment 373  
shell procedures for accounting 31  
shell variable arithmetic 317  
shift command 656  
shlib command 660-662  
show login records 33  
showing  
  a calendar 106  
  accounting report 33  
  compressed files 543  
  corresponding group names and IDs 395  
  corresponding user names and IDs 395  
  current directory 589  
  date 219  
  documents formatted with the Memorandum  
  Macros 492  
  file checksum 726  
  formatted files 553, 561  
  login name 456  
  news items 512  
  packed files 543  
  process accounting records 38  
  process status 579  
  profile data 571  
  SCCS file editing activity 609  
  session record 33  
  squeezed files 543  
  total accounting report 34  
shutacct command 34  
shutdown command 663-664  
shutting down the system 663  
siline command 701  
sin command 702  
size command 665-666  
skulker command 667  
sleep command 668-669  
sno command 670-671  
sort command 672-678  
sorting files 672  
sound command 679-680  
source code  
  external references flow graph 125  
Source Code Control System  
  See SCCS  
special (device) files  
  /dev/null  
  acctcom 38  
  standard input assigned to 38  
  adding 241  
  changing 241  
  creating 490  
  deleting 241  
special user ID  
  adm 606  
  root 422, 606, 724, 725  
specifying version date cutoff 364

---

spell command 681-683  
 spellin command 682  
 spellprog program 683  
 spline command 684-685  
 split command 686  
 split files by context 202  
 splitting a file into pieces 686  
 splp command 687-689  
 squeezing files 543  
 standard input  
   acctcon1 42  
   acctmerg 46  
   as 64  
   at 66  
   awk 70  
   batch 66  
   bdiff 88  
   cb 111  
   cmp 138  
   comb 142  
   fwtmp 345  
   wtmpfix 346  
 standard output  
   acctcon1 42  
   acctmerg 46  
   acctwtmp 345  
   awk 70  
   bdiff 88  
   cal 106  
   cb 111  
   cflow 125  
   cmp 138  
   comb 142  
   fwtmp 345  
   wtmpfix 346  
 start accounting functions 34  
 starting up the system 396  
 startup command 34  
 startup initialization 594  
 startup shell  
   actman 50  
 stat commands 690-715  
 state values  
   changing 272.4  
 status of inter-process communication 411  
 stop a process 422  
 stopping error-logging demon 309  
 stream editor 629  
 strings 80  
 strip command 716  
 stty command 717-723  
 su command 724-725  
 subset command 702  
 sum command 726  
 summarize disk usage 273  
 summary of command usage 36  
 superblock update 727  
 superuser authority 568, 569, 737, 811  
   commands  
     adduser 802  
     at 66  
     chmod 128  
     chroot 134  
     ckprereq 406  
     cpio 159  
     crontab 174  
     cvid 212  
     date 219  
     devices 241  
     dfscck 335  
     errdemon 303  
     errstop 309  
     fsck 333  
     installp 402  
     killall 425  
     mesg 484  
     minidisks 485  
     mount 498  
     mvmd 407  
     nice 515  
     passwd 546  
     print 568  
     removing scheduled jobs 67  
     rm 601  
     shown in report 39  
     shutdown 663  
     tar 736  
     updatep 796  
     users 802  
     uustat 810  
   dsipc 272.1  
   dslcxprof 272.2

---

dsstate 272.4  
 dsxlate 272.6  
 iptable 414.1  
 ndtable 506.1  
 removing files 601  
 uhtable 784  
 suspend a process 422  
 sync command 727  
 syntax checking, C programs 446  
 system activity graph 612  
 system activity reporter 610, 614  
 system files  
   \$HOME/.profile 725  
   /dev/null  
     expr 320  
   /etc/.ilog 454, 851  
   /etc/budate  
     mount 78  
   /etc/environment 398, 454  
   /etc/filesystems 334, 344, 487  
     cvid 212  
     mount 77, 498, 499  
   /etc/group 802  
   groups 385  
   /etc/magic 324  
   /etc/mnttab 499  
     mount 498  
     umount 786  
   /etc/ogroup 802  
   /etc/opasswd 802  
   /etc/passwd 454, 724, 802  
     acctdisk 44  
     acctprcl 48  
     groups 385  
   /etc/portstatus 397, 551  
   /etc/profile 725  
   /etc/qconfig 558  
   /etc/rasconf 303  
   /usr/adm/sulog 724  
   /usr/adm/user.cfile 802, 804  
   /usr/adm/wtmp 346, 851  
   /usr/games/lib/backrules 75  
   /usr/lib/cron/at.allow 66  
   /usr/lib/cron/at.deny 66  
   /usr/lib/eign 585  
   /usr/lib/terminfo 555  
   /usr/lib/tmac 239  
   a.out 64  
   connect.con 152  
   group 327, 510  
   passwd 327, 510, 546  
   ports 550, 851  
   portstatus 550  
   utmp 851  
 system group commands  
   adduser 802  
   backup 76  
   ckprereq 406  
   cvid 212  
   date 219  
   devices 241  
   dsipc 272.1  
   dsldxprof 272.2  
   dsstate 272.4  
   dsxlate 272.6  
   installp 402  
   iptable 414.1  
   minidisks 485  
   mount 498  
   mvmd 407  
   ndtable 506.1  
   print 568  
   turnoff 783  
   turnon 783  
   uhtable 784  
   users 802  
 system image examination 168  
 system parameters, changing or examining 133  
 system procedures  
   starting the error-logging demon 303  
 system startup 396

**T**

tab command 728  
 tabs command 729-731  
 tail command 732-733  
 tape archiver 735  
 tapechk command 734  
 tar command 735-738

---

tbl command 739, 741  
  descriptions removed by the deroff  
  command 239  
  used in pipeline with nroff 526  
tc command 742-743  
tctl command 744-745  
td command 348  
tee command 746-747  
tekset command 348  
termdef command 748  
terminal characteristics 748  
terminal mapping 721  
terminals  
  DASI 300 863  
  DASI 300s 863  
  DASI 450 866  
  Diablo 1620 866  
  HP2621 392  
  HP2640 392  
  phototypesetter simulator 742  
  Tektronix 4014 742, 865  
  Xerox 1700 866  
terminals, multiple virtual 50  
terminating error-logging demon 309  
test command 656, 750-752  
text file, changing the format of 507  
text processing commands  
  checkcw 213  
  checkeq 300  
  checkmm 492  
  col 140  
  cw 213  
  eqn 300  
  eqncheck 300  
  greek 379  
  mant 495  
  mm 492  
  mmt 495  
  mt 495  
  mvt 495  
  neqn 300  
  nroff 525  
  spell 681  
  tbl 739  
  troff 526  
tic command 753  
time a command 754, 755  
time command 754  
TIMEOUT 647  
times command 657  
timex command 755-756  
title command 715  
toc commands 757-759  
total command 708  
touch command 760-761  
tplot command 762  
tput command 763-764  
tr command 765-767  
trace command 768-771  
translate characters 765  
translate profile  
trap command 657  
trcrpt command 772-773  
trcstop command 774  
trcupdate command 775-776  
troff command 526, 536  
  requests removed by the deroff  
  command 239  
  tbl, preprocessor 739  
true command 777  
tsort command 778-779  
ttoc command 758  
tvt game 780  
tty command 781-782  
turn off process accounting 34  
turn on accounting functions 34  
turnacct command 34  
turning the computer off 663  
turnoff command 783  
turnon command 783  
type command 657

**U**

ugtable command 784  
ulimit command 657  
umask command 657, 784.1-784.2  
umount command 499, 786-787  
uname command 788-789  
unget command 790-791



---

uniq command 792  
units command 793-795  
unlink 444  
unlink command 445  
unmount command 786-787  
unmounting a file system 786  
unpack command 543-545  
unset command 657  
untab command 728  
until command 654  
update a program 800  
update groups of programs 474  
updatep command 796-801  
updating access times of a file 760  
updating modification times of a file 760  
updating the superblock 727  
usage, command summary 36  
user  
    adding 802  
    adm 33  
    changing 802  
    deleting 802  
user ID, special  
    adm 606  
    root 422, 606, 724, 725  
user IDs and names, displaying 395  
user mask, modifying 784.1  
users command 802-804  
using  
    display station features 258  
uuclean command 805-806  
uucp command 807-809  
uulog command 807-809  
uname command 807-809  
uupick command 815-817  
uustat command 810-812  
uusub command 813-814  
uuto command 815-817  
uux command 818-820

## V

val command 821-822  
var command 709  
varyon command 823-825  
vc command 826-829  
vcc command 113  
vedit command 832-841  
verify command 830-831  
version date cutoff, specifying 364  
vi command 832-841  
view command 832-841  
virtual terminal  
    assigning default display 258  
    assigning physical display 258  
    changing DMA pinned page 259  
    setting active color palette 258  
    setting background colors 258  
    setting fonts 258  
    setting foreground colors 258  
virtual terminal manager 50  
vrmconfig command 842-843  
vrmfmt command 113  
vtoc command 758

## W

wait command 657, 844  
wall command 845  
wc command 846-847  
what command 848-849  
whatis command 388  
    utility commands 389  
while command 653  
who command 850-852  
work station characteristics 748  
work station parameters, setting 717  
work stations  
    DASI 300 863  
    DASI 300s 863  
    DASI 450 866  
    Diablo 1620 866

---

HP2621 392  
HP2640 392  
phototypesetter simulator 742  
Tektronix 4014 742, 865  
Xerox 1700 866  
write command 853-855  
write operations 40  
writing buffered files to fixed disk 727  
writing the last part of a file 732  
writing to standard output  
  acctcon1 42  
  acctmerg 46  
  acctwtmp 345  
  awk 70  
  bdiff 88  
  cal 106  
  cb 111  
  cflow 125  
  cmp 138  
  comb 142  
  fwtmp 345  
  wtmpfix 346

wtmpfix command 346  
wump game 856

**X**

xargs command 857-860

**Y**

yacc command 861-862

**Numerics**

300 command 863-864  
4014 command 865  
450 command 866-867





The IBM RT PC  
Programming Family

## **Reader's Comment Form**

**AIX Operating System  
Commands Reference**

SC23-0790-0

Your comments assist us in improving our products. IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

For prompt resolution to questions regarding set up, operation, program support, and new program literature, contact the authorized IBM RT PC dealer in your area.

Comments:



NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES

# BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation  
Department 997, Building 998  
11400 Burnet Rd.  
Austin, Texas 78758



Fold and tape

fold and tape

Cut or Fold Along Line

addressee

addressee

Book Title \_\_\_\_\_

Order No. \_\_\_\_\_

**Book Evaluation Form**

Your comments can help us produce better books. You may use this form to communicate your comments about this book, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Please take a few minutes to evaluate this book as soon as you become familiar with it. Circle Y (Yes) or N (No) for each question that applies and give us any information that may improve this book.

Y N Is the purpose of this book clear?  
\_\_\_\_\_  
\_\_\_\_\_

Y N Are the abbreviations and acronyms understandable?  
\_\_\_\_\_  
\_\_\_\_\_

Y N Is the table of contents helpful?  
\_\_\_\_\_  
\_\_\_\_\_

Y N Are the examples clear?  
\_\_\_\_\_  
\_\_\_\_\_

Y N Is the index complete?  
\_\_\_\_\_  
\_\_\_\_\_

Y N Are examples provided where they are needed?  
\_\_\_\_\_  
\_\_\_\_\_

Y N Are the chapter titles and other headings meaningful?  
\_\_\_\_\_  
\_\_\_\_\_

Y N Are the illustrations clear?  
\_\_\_\_\_  
\_\_\_\_\_

Y N Is the information organized appropriately?  
\_\_\_\_\_  
\_\_\_\_\_

Y N Is the format of the book (shape, size, color) effective?  
\_\_\_\_\_  
\_\_\_\_\_

Y N Is the information accurate?  
\_\_\_\_\_  
\_\_\_\_\_

**Other Comments**

What could we do to make this book or the entire set of books for this system easier to use?  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

Y N Is the information complete?  
\_\_\_\_\_  
\_\_\_\_\_

Y N Is only necessary information included?  
\_\_\_\_\_  
\_\_\_\_\_

Y N Does the book refer you to the appropriate places for more information?  
\_\_\_\_\_  
\_\_\_\_\_

**Optional Information**

Y N Are terms defined clearly?  
\_\_\_\_\_  
\_\_\_\_\_

Your name \_\_\_\_\_

Company name \_\_\_\_\_

Street address \_\_\_\_\_

Y N Are terms used consistently?  
\_\_\_\_\_  
\_\_\_\_\_

City, State, ZIP \_\_\_\_\_



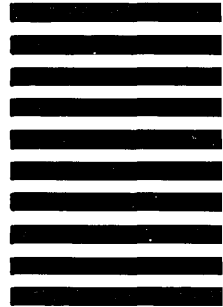
NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES

**BUSINESS REPLY MAIL**

FIRST CLASS PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation  
Department 997, Building 998  
11400 Burnet Rd.  
Austin, Texas 78758



Fold and tape

fold and tape

Cut or Fold Along Line

adp i

adp i

adp i