ERRATA

09-02486

The attached pages update the following:

CTIX Operating System Manual,
Version C, Volume 2,
2nd Edition
09-02263-01

CTIX Operating System Manual,
Version C, Volume 3,
2nd Edition
09-02264-01

- The man page for *masterupd*(1M) should be removed from Volume 2, and all references to this command in the manuals should be ignored.
- The *route*(1M) man page included with this errata should be substituted for the *route*(1M) man page in Volume 2.
- Insert the *accept*(2) man page, included with this errata, in Section 2 of Volume 3, immediately after the *intro*(1M) man page.

NAME

route - manually manipulate the routing tables

SYNOPSIS

/etc/route [**-f**] [command destination gateway [metric]]

DESCRIPTION

route is a program used to manually (statically) manipulate the network routing tables. It is normally not needed, since the routing daemon, *routed* manages the system routing table and therefore handles this function. Static routing is generally appropriate in cases where packets are being sent to systems that do not recognize broadcast routing packets.

Only the super-user may modify routing tables.

route accepts two commands: *add*, to add a route; and *delete*, to delete a route.

All commands have the following syntax:

/etc/route *command destination gateway* [*metric*]

where *destination* is a host or network for which the route is "to," *gateway* is the gateway to which packets should be addressed, and *metric* indicates the "hop count," the number of interfaces a packet must pass through to reach its *destination*. If no metric is specified, *route* assumes a value of 0. Routes to a particular host are distinguished from those to a network by interpreting the Internet address associated with *destination*. All symbolic names specified for a destination or gateway are first looked up in the host name database. If this lookup fails, *route* looks for the name in the network name database.

If the **-f** option is specified, *route* will "flush" (delete) the routing tables of all gateway entries.

DIAGNOSTICS

add host*: gateway* host *flags* hex-flag

The specified route is being added to the tables. The values printed are from the routing table entry supplied in the *ioctl* call.

delete host*: gateway* host *flags* hex-flags

As above, but when deleting an entry.

host host *done*

When the **-f** flag is specified, each routing table entry is deleted. When a deletion is done by host name, an error is indicated with a message of this form.

not in table

A delete operation was attempted without the **-f** flag for a host name that is not present in the route table.

routing table overflow

An add operation was attempted, but the system was low on resources and was unable to allocate memory to create the new entry.

SEE ALSO

intro(4), adman(1), routed(1M), hosts(4), networks(4).

NAME

accept - accept a connection on a socket

SYNOPSIS

```
#include <sys/types.h>
#include <sys/socket.h>

int accept(s, addr, addrlen)
int s;
struct sockaddr *addr;
int *addrlen;
```

DESCRIPTION

The *accept* call accepts a connection on a socket. The argument *s* is a socket that has been created with *socket*(2), bound to an address with *bind*(2), and is listening for connections after a *listen*(2). The *accept* extracts the first connection on the queue of pending connections, creates a new socket with the same properties of *s* and allocates a new file descriptor for the socket. If no pending connections are present on the queue, and the socket is not marked as non-blocking, *accept* blocks the caller until a connection is present. If the socket is marked non-blocking and no pending connections are present on the queue, *accept* returns an error as described below. The accepted socket, *ns*, cannot be used to accept more connections. The original socket *s* remains open.

The argument *addr* is a result parameter which is filled in with the address of the connecting entity, as known to the communications layer. The exact format of the *addr* parameter is determined by the "communications domain" [see *protocols*(4)]. The *addrlen* is a value-result parameter; it should initially contain the amount of space pointed to by *addr*; on return it contains the actual length (in bytes) of the address returned. This call is used with connection-based socket types, currently with SOCK_STREAM.

RETURN VALUE

The call returns -1 on error. If it succeeds it returns a non-negative integer which is a descriptor for the accepted socket (*ns*, described above).

ERRORS

The *accept* will fail if:

[EBADF]	The descriptor is invalid.
[ENOTSOCK]	The descriptor references a file, not a socket.
[EOPNOTSUPP]	The referenced socket is not of type SOCK_STREAM.
[EFAULT]	The *addr* parameter is not in a writable part of the user address space.

SEE ALSO

bind(2), connect(2), intro(2), listen(2), socket(2), intro(7).
CTIX Network Programmer's Primer.

The enclosed pages update the following manual:

CTIX Operating System Manual,
Version C
Second Edition
09-02263-01

Insert the pages of this Update Notice into the *CTIX Operating System Manual, Version C* according to the instructions below. Each new page is dated so that you know which pages are update pages and which are original pages. The Summary of Changes on the first page of this Update Notice summarizes the changes made.

All material discussed in this Update Notice will be incorporated into the next edition of the *CTIX Operating System Manual, Version C*. To retain a record of this Update Notice, insert this cover page immediately after the manual's title page.

For Volume 1:

Find and Remove:	**Replace With:**
title page	title page
iii to vii	iii to viii.b
xxix to xciii	xxix to xciii
adb(1)	adb(1)
as(1)	as(1)
cc(1)	cc(1)
cc1sw(1)	cc1sw(1)
config(1M)	config(1M)
cpp(1)	cpp(1)
crash(1M)	crash(1M)
createdev(1M)	createdev(1M)
fsck(1M)	fsck(1M)
fstyp(1M)	fstyp(1M)
hinv(1)	hinv(1)
includes(1)	includes(1)
-----	iopdump(1M)
iv(1)	iv(1)

For Volume 2:

Find and Remove:	**Replace With:**
title page	title page
iii to vii	iii to vii
xiii to lxxvii	-----
machid(1)	machid(1)
masterupd(1M)	-----
mcs(1)	mcs(1)
mkfs(1M)	mkfs(1M)
mount(1M)	mount(1M)
-----	occ(1)
passmgmt(1M)	passmgmt(1M)
passwd(1)	passwd(1)
profiler(1M)	profiler(1M)
pwconv(1M)	pwconv(1M)
sar(1)	sar(1)
sdb(1)	sdb(1)
serstat(1M)	serstat(1M)
tio(1)	tio(1)

For Volume 3:

Find and Remove:	Replace With:
title page	title page
iii to vii	iii to vii
xiii to lxxvii	-----
gettimeofday(2)	gettimeofday(2)
notify(2)	notify(2)
shmop(2)	shmop(2)
syslocal(2)	syslocal(2)
fpgetround(3C)	fpgetround(3C)
getspent(3X)	getspent(3X)
monitor(3C)	monitor(3C)
sleep(3C)	sleep(3C)

For Volume 4:

Find and Remove:	Replace With:
title page	title page
iii to v	iii to v
xi to lxxv	-----
filehdr(4)	filehdr(4)
passwd(4)	passwd(4)
shadow(4)	-----
-----	timezone(4)
disk(7)	disk(7)

CTIX™ OPERATING SYSTEM MANUAL

Version C
Volume 1

This manual was prepared on a Convergent Technologies S/640 Computer System and was printed on an Apple LaserWriter II Laser Printer.

Second Edition (November 1989) 09-02262-01
Update Notice 1 (November 1990) 09-02578

TABLE OF CONTENTS: VOLUME 1

UPDATE NOTICE 1 SUMMARY OF CHANGES

Update Notice 1 to the *CTIX Operating System Manual, Version C, Second Edition*, documents the new commands and features of the 6.3 release of CTIX, which runs on the new S/4040, a Motorola 68040-based system. Changes to the manual are summarized below.

Volume 1

- Revised front matter
- New pages:

  iopdump(1M).
- Revised pages:

  adb(1), *as*(1), *cc*(1), *cc1sw*(1), *config*(1M), *cpp*(1), *crash*(1M), *createdev*(1M), *fsck*(1M), *fstyp*(1M), *hinv*(1), *includes*(1), *iv*(1).

Volume 2

- Revised front matter
- New pages:

  occ(1).
- Revised pages:

  machid(1), *mcs*(1), *mkfs*(1M), *mount*(1M), *passmgmt*(1M), *passwd*(1), *profiler*(1M), *pwconv*(1M), *sar*(1), *sdb*(1), *serstat*(1M), *tio*(1).
- Deleted pages:

  masterupd(1M).

Volume 3

- Revised pages:

  gettimeofday(2), *notify*(2), *shmop*(2), *syslocal*(2), *fpgetround*(3C), *getspent*(3X), *monitor*(3C), *sleep*(3C).

Volume 4

- Revised front matter
- New pages:
  timezone(4).
- Revised pages:
  filehdr(4), *passwd*(4), *disk*(7).
- Deleted pages:
  shadow(4).

PERMUTED INDEX

This index includes entries for all pages of Volumes 1 through 4. The entries themselves are based on the one-line descriptions or titles found in the NAME portion of each manual page; the significant words (keywords) of these descriptions are listed alphabetically down the center of the index.

The index is actually a keyword-in-context (KWIC) index that has three columns. To use the index, read the center column to look up specific commands by name or by subject topics. Note that the entry may begin in the left column or wrap around and continue into the left column. A period (.) marks the end of the entry, and a slash (/) indicates where the entry has been continued or truncated. The right column gives the manual page where the command or subject is described.

NAME

adb - absolute debugger

SYNOPSIS

adb [**-w**] [objfil [corfil]]

DESCRIPTION

The *adb* program is a general-purpose debugging program. It can be used to examine files and to provide a controlled environment for the execution of CTIX programs.

The *objfil* parameter is normally an executable program file, preferably containing a symbol table; if not, the symbolic features of *adb* cannot be used, but the file can still be examined. The default for *objfil* is **a.out**. The *corfil* parameter is assumed to be a core image file produced after executing *objfil*; the default for *corfil* is **core**.

Requests to *adb* are read from the standard input and responses are to the standard output. If the **-w** flag is present, both *objfil* and *corfil* are created, if necessary, and opened for reading and writing so that files can be modified using *adb*. Note that *adb* ignores QUIT; INTERRUPT causes return to the next *adb* command.

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

[*address*] [, *count*] [*command*] [;]

If *address* is present, *dot* is set to *address*. Initially, *dot* is set to 0. For most commands, *count* specifies how many times the command is executed. The default *count* is 1. *Address* and *count* are expressions.

The interpretation of an address depends on the context it is used. If a subprocess is being debugged, then addresses are interpreted in the usual way in the address space of the subprocess. For further details of address mapping, see *ADDRESSES*.

EXPRESSIONS

. The value of *dot*.

\+ The value of *dot* incremented by the current increment.

^ The value of *dot* decremented by the current increment.

" The last *address* typed.

integer Hexadecimal by default or if preceded by 0x; octal if preceded by 0o or 0O; decimal if preceded by 0t or 0T.

integer.fraction
A 32-bit floating-point number.

'cccc' The ASCII value of up to four characters. A \ may be used to escape a '.

<name The value of *name*, which is either a 68010/68020/68040 register name or a variable name. *adb* maintains a number of variables (see *VARIABLES*) named by single letters or digits. If *name* is a register name, then the value of the register is obtained from the system header in *corfil*. The registers are **d0** through **d7**, **a0** through **a7**, **sp**, **pc**, **cc**, **sr**, and **usp**.

symbol A *symbol* is a sequence of upper or lowercase letters, underscores, or digits, not starting with a digit. The value of the *symbol* is taken from the symbol table in *objfil*.

From C, only external variables are available as symbols. The symbol name is the same as the C variable name, except that an underscore (_) is prepended to any name that is the same as the name for a register.

(*exp*) The value of the expression *exp*.

Monadic operators:

**exp* The contents of the location addressed by *exp* in *corfil*.

@exp The contents of the location addressed by *exp* in *objfil*.

-exp Integer negation.

~exp Bitwise complement.

Dyadic operators are left associative and are less binding than monadic operators.

e1+e2 Integer addition.

e1-e2 Integer subtraction.

*e1*e2* Integer multiplication.

e1%e2 Integer division.

e1&e2 Bitwise conjunction.

e1|e2 Bitwise disjunction.

e1#e2 *E1* rounded up to the next multiple of *e2*.

COMMANDS

Most commands consist of a verb followed by a modifier or list of modifiers. The following verbs are available. (The commands **?** and **/** may be followed by *****; see *ADDRESSES* for further details.)

?*f* Locations starting at *address* in *objfil* are printed according to the format *f*. The value of *dot* is incremented by the sum of the increments for each format letter (q.v.).

/*f* Locations starting at *address* in *corfil* are printed according to the format *f* and *dot* is incremented as for **?**.

=*f* The value of *address* is printed in the styles indicated by the format *f*. (For **i** format **?** is printed for the parts of the instruction that reference subsequent words.)

A *format* consists of one or more characters that specify a style of printing. 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* is incremented by the amount given for each format letter. If no format is given, then the last format is used. The format letters available are as follows:

o 2 Prints 2 bytes in octal. All octal numbers output by *adb* are preceded by 0.

O 4 Prints 4 bytes in octal.

q 2 Prints in signed octal.

Q 4 Prints long signed octal.

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.

f 4 Prints the 32-bit value as a floating-point number.

F 8 Prints double floating point.

b 1 Prints the addressed byte in octal.

c 1 Prints the addressed character.

C 1 Prints the addressed character using the following escape convention. Character values 000 to 040 are printed as **@**, followed by the corresponding character in the range 0100 to 0140. The character **@** is printed as **@@**.

s *n* Prints the addressed characters until a zero character is reached.

S *n* Prints a string using the @ escape convention. The value *n* is the length of the string including its zero terminator.

Y 4 Prints 4 bytes in date format [see *ctime*(3C)].

i n Prints as machine instructions. The value *n* is the number of bytes occupied by the instruction. This style of printing causes variables 1 and 2 to be set to the offset parts of the source and destination, respectively.

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 2 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 eight-space tab stop.

r 0 Prints a space.

n 0 Prints a new-line.

"..." 0 Prints the enclosed string.

^ The value of *dot* is decremented by the current increment. Nothing is printed.

\+ The value of *dot* is incremented by 1. Nothing is printed.

\- The value of *dot* is decremented by 1. Nothing is printed.

new-line

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

[?/]**l** *value 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.

[?/]**w** *value* ...

Writes the 2-byte *value* into the addressed location. If the command is **W**, write 4 bytes. Odd addresses are not allowed when writing to the subprocess address space.

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

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

>*name* The value of *dot* is assigned to the variable or register named.

! A shell is called to read the rest of the line following **!**.

$*modifier*

Miscellaneous commands. The available *modifiers* follow:

- <*f* Reads commands from the file *f* and return.
- >*f* Sends output to the file *f*, which is created if it does not exist.
- **r** Prints the general registers and the instruction addressed by **pc**. The value of *dot* is set 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 **fp**). If *count* is given, then only the first *count* frames are printed.
- **e** The names and values of external variables are printed.
- **w** Sets the page width for output to *address* (default 80).
- **s** Sets the limit for symbol matches to *address* (default 255).
- **o** All integers input are regarded as octal.
- **d** Resets integer input as described in *EXPRESSIONS*.
- **q** Exits from *adb*.

v Prints all nonzero variables.

f Prints the 68881 floating-point registers.

m Prints the address map.

:*modifier*

Manage a subprocess. Available modifiers are as follows:

b*c* Sets breakpoint at *address*. The breakpoint is executed *count*-1 times before causing a stop. Each time the breakpoint is encountered, the command *c* is executed. If this command sets *dot* to zero, the breakpoint causes a stop.

d Deletes breakpoint at *address*.

r Runs *objfil* as a subprocess. If *address* is given explicitly, then the program is entered at this point; otherwise, the program is entered at its standard entry point. The value *count* specifies how many breakpoints are to be ignored before stopping. Arguments to the subprocess can 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. All signals are turned on entry to the subprocess.

c*s* The subprocess is continued with signal *s* [see *signal*(2)]. If *address* is given, the subprocess is continued at this address. If no signal is specified, then the signal that caused the subprocess to stop is sent. Breakpoint skipping is the same as for **r**.

s*s* As for **c** except that the subprocess is single-stepped *count* times. If there is no current subprocess, then *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 The current subprocess, if any, is terminated.

VARIABLES

The *adb* command provides a number of variables. Named variables are set initially by *adb*, but they are not used subsequently. Numbered variables are reserved for communication as follows.

0 The last value printed.

1 The last offset part of an instruction source.

2 The previous value of variable 1.

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

b The base address of the data segment.

d The data segment size.

e The entry point.

m The "magic" number (0407, 0410, or 0413).

s The stack segment size.

t The text segment size.

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*) and the *file address* corresponding to a written *address* is calculated as follows:

b1≤*address*<*e1* => *file address*=*address*+*f1*-*b1*

otherwise,

b2≤*address*<*e2* => *file address*=*address*+*f2*-*b2*,

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

So you can use *adb* on large files, all appropriate values are kept as signed 32-bit integers.

FILES

/dev/kmem
/dev/swap
a.out
core

SEE ALSO

ptrace(2), a.out(4), core(4).

DIAGNOSTICS

adb when there is no current command or format. Comments about inaccessible files, syntax errors, abnormal termination of commands, and so on. Exit status is 0, unless last command failed or returned nonzero status.

BUGS

A breakpoint set at the entry point is not effective on initial entry to the program.

When single-stepping, system calls do not count as an executed instruction.

Local variables whose names are the same as an external variable may foul up the accessing of the external.

Shared libraries are not included in the map.

NAME

as - common assembler

SYNOPSIS

as [options] filename

DESCRIPTION

The *as* command assembles the named file. The following flags may be specified in any order:

-o *objfile* Puts the output of the assembly in *objfile*. By default, the output filename is formed by removing the **.s** suffix, if there is one, from the input filename and appending a **.o** suffix.

-n Turns off long/short address optimization. By default, address optimization takes place.

-m Runs the *m4* macro processor on the input to the assembler.

-R Removes (unlink) the input file after assembly is completed.

-dl Does not produce line number information in the object file.

-T Truncates symbols to eight characters.

-V Writes the version number of the assembler being run on the standard error output.

-Y *[md],dir* Finds the **m4** preprocessor (**m**) and/or the file of predefined macros (**d**) in directory *dir* instead of in the customary place.

FILES

*TMPDIR/** temporary files

TMPDIR is usually **/tmp** but can be redefined by setting the environment variable **TMPDIR** [see *tempnam*( ) in *tmpnam*(3S)].

SEE ALSO

cc(1), ld(1), m4(1), nm(1), strip(1), tmpnam(3S), a.out(4).
Programmer's Guide: CTIX Supplement.

WARNING

If the **-m** (*m4* macro processor invocation) option is used, keywords for *m4* [see *m4*(1)] cannot be used as symbols (variables, functions, or labels) in the input file since *m4* cannot determine which are assembler symbols and which are real *m4* macros.

CAVEATS

Arithmetic expressions may only have one forward referenced symbol per expression.

NOTES

Wherever possible, the assembler should be accessed through a compilation system interface program [such as *cc*(1)].

NAME

cc - C compiler

SYNOPSIS

cc [options] files

DESCRIPTION

The *cc* command is the interface to the C Compilation System. The compilation tools consist of a preprocessor, optimizing compiler, assembler, and link editor. The *cc* command processes the supplied options and then executes the various tools with the proper arguments. The *cc* command accepts several types of files as arguments.

Files whose names end with **.c** are taken to be C source programs and may be preprocessed, compiled, optimized, assembled, and link edited. The compilation process may be stopped after the completion of any pass if the appropriate options are supplied. If the compilation process runs through the assembler, then an object program is produced and is left in the file whose name is that of the source with **.o** substituted for **.c**. However, the **.o** file is normally deleted if a single C program is compiled and then immediately link edited. In the same way, files whose names end in **.s** are taken to be assembly source programs, and may be assembled and link edited; files whose names end in **.i** are taken to be preprocessed C source programs and may be compiled, optimized, assembled, and link edited. Files whose names do not end in **.c, .s,** or **.i** are handed to the link editor.

Since the *cc* command usually creates files in the current directory during the compilation process, it is necessary to run the *cc* command in a directory in which a file can be created. The following options are interpreted by *cc*.

-#

-## These options cause *cc* to display each command that it would generate if it were to execute, but to fully execute only in the case of **-#**. Thus, **-#** specifies execution in verbose mode, and **-##** specifies verbose mode (what *cc* would do if it were to execute), but does nothing.

-c Suppresses the link editing phase of the compilation, and does not remove any produced object files.

-g Causes the compiler to generate additional information needed for the use of *sdb* (1).

-o *outfile*

Produces an output object file by the name *outfile*. The name of the default file is **a.out**. This is a link editor option.

-p Arranges for the compiler to produce code that counts the number of times each routine is called; also, if link editing takes place, profiled versions of libc.a and libm.a (with -lm option) are linked and *monitor*(3C) is automatically called. A **mon.out** file will then be produced at normal termination of execution of the object program. An execution profile can then be generated by use of *prof*(1).

-w Suppresses warnings. Previous versions of the compiler passed the **-w** option to the linker (*ld*) to suppress warnings from *ld* about truncated names. Use **-wl,-w** instead.

-B*string*
-t[p02al]
These options will be removed in the next release. Use the **-Y** option.

-E Runs only *cpp*(1) on the named C programs, and sends the result to the standard output.

-H Prints out on *stderr* the pathname of each file included during the current compilation.

-O Does compilation phase optimization. This option will not have any affect on **.s** files.

-O*n* Optimizes code, where *n* is a decimal number specifying the optimizations to be done:

- **2** Branch optimization. (Jump to jump, and so forth.)
- **4** Strength reduction, like replacing constant multiplications with shifts and adds.
- **8** Reaching analysis. Used by common subexpression eliminations, constant propagations, loop optimizations, and so forth.
- **16** Loop optimizations. Move loop-invariants out of loop, replace array references with pointers, invert loop-index and count to zero, loop unrolling, and so forth.
- **32** Lifetime analysis. Lets variables share registers.
- **64** Remove useless code, like unreached statements and assignments to unused variables.
- **128** Optimize more than one function at a time. See **-Xparse-size (-X20)**.
- **256** Remove link/unlink, if possible.

Other optimizations, like peephole and automatic register allocation are always done.

Optimizations can be combined by adding their codes; for example, to have both strength reduction and reaching analysis, *n* would be specified as **12**. **-O** without a number implies **-O511**, which is a combination of all optimizations listed above. If **-O** is not given, then **-O6** is implied.

-P Runs only *cpp*(1) on the named C programs and leaves the result in corresponding files suffixed **.i**. This option is passed to *cpp*(1).

-S Compiles and does not assemble the named C programs, and leaves the assembler-language output in corresponding files suffixed **.s**.

-T Truncates variable names to eight characters. Tells the loader to match eight character names (same as **-G** in the loader).

-V Displays the current version number.

-Wc, arg1[, arg2...]
Hands off the argument[s] *argi* to pass *c* where *c* is one of **[p0al]** indicating the preprocessor, compiler, assembler, or link editor, respectively. For example: **-Wa,-m** passes **-m** to the assembler.

-X Since the descriptions for these options take up so many pages, the **-X** options are listed after the **-Y** options below.

-Y [p0alSILUc], *dirname* | *processor*
Specifies a new pathname, *dirname*, for the locations of the tools and directories designated in the first argument; or selects a processor type, *processor*, for which to generate code. **[p0alSILUc]** represents:

- **p** preprocessor
- **0** compiler
- **a** assembler
- **l** link editor
- **S** directory containing the start-up routines
- **I** default include directory searched by *cpp*(1)
- **L** first default library directory searched by *ld*(1)
- **U** second default library directory searched by *ld*(1)
- **c** selects the processor type, specified by the second argument, for which to generate code: **68040**, **68020**, **68010**, **68881**. For example, **-Y c,68020** selects the 68020 processor with software floating-point instructions. Note that **68881** implies **68020**, and that if a compile is being done for a 68000- or 68010-based system, *occ* is called automatically.

If the location of a tool is being specified, then the new pathname for the tool will be *dirname/tool*. If more than one **-Y** option is applied to any one tool or directory, then the last occurrence holds.

-X*name*

-X*n* The **-X** switch tells the compiler how to behave in certain areas. There are two ways to set these options, either by a number, **-X***n*, or with a mnemonic, **-X***name*.

Some options can be set to a decimal value, *m*, by using an equals sign, **-X***n*=*m* or **-X***name*=*m*.

To turn off an option, set it to zero: **-X***n*=**0** or **-X***name*=**0**. If nothing else is said, the default value of *m* is 1. For example, **-X6**, **-X6=1**, and **-Xtest-at-top** are the same option.

Currently the following **-X** options are implemented:

-Xstruct-compress

-X1=1 Uses same alignment in structure as the member with the biggest alignment requirement. On systems where the minimum structure alignment is always bigger than 1, this option can be used to match external hardware.

-Xstruct-unaligned

-X1=2 (Default) Does not align structures when pushing them on the stack as arguments.

-Xstruct-default

-X1=0 Uses default structure alignment.

-Xmismatch-warning

-X2 (Default) Generates only a warning instead of a fatal error when pointers of different types or integers are mixed in expressions. For example, the line

```
long i1, i2 = &i1;
```

is illegal in the C programming language, but some older programs depend on the compiler to handle lines of code like this, anyway. **-X2** is also set by **-Xpcc** (**-X7=3**).

-Xuse-float

-X3 Forces the compiler not to convert float operands to double in expressions and as function arguments.

-Xmemory-is-volatile

-X4 Does not do optimizations that can make drivers (or other programs) fail. By default, the compiler tries to store as much data in registers as is possible, whenever it is safe. Problems occur if a memory location changes because it is mapped to some external hardware and the compiler, unaware of this change, continues to use the old value that is stored in a register. These situations can be handled with the ANSI-C keyword **volatile**, but to be able to compile older programs, **-Xmemory-is-volatile** is provided.

-Xlocals-on-stack

-X5 By default, the compiler tries to allocate all local variables to registers. If **-Xlocals-on-stack** is given, only variables declared with the register keyword are assigned to registers.

-Xtest-at-bottom

-X6=0 Uses one loop test at the bottom of a loop. The default is X6=1.

-Xtest-at-top

-X6=1 (Default) Uses one loop test at the top of a loop.

-Xtest-at-both

-X6=2 Forces the compiler to always test loops both before the loop is started and at the bottom of the loop. Use this option to create the fastest possible code, although using somewhat more space. Even if **-Xtest-at-both** is not set, other optimizations sometimes make the compiler generate double tests.

-Xk-and-r

-X7=0 With this option, the compiler follows the C language standard as defined by the Kernighan & Ritchie C reference manual, but with all new ANSI-C features added. Where Kernighan & Ritchie and ANSI differs, **-Xk-and-r** follows Kernighan & Ritchie. See Table 1, later in this section, which explains the cases where the **-X7** switch has an effect. The default is X7=3.

-Xpcc

-X7=3 (Default) With this option, the compiler follows the C standard as defined by the Unix System V Release 3 C compiler. See Table 1 for details.

-Xenum-is-small
-X8=0 Uses the smallest integer type possible for **enum**s.

-Xenum-is-int
-X8 (Default) The **enum** type is always equal to **int**. This option is also set by the **-Xpcc** (**-X7=3**) option.

-Xforce-declarations
-X9 With this option, the compiler generates warnings if a function is used without a previous declaration. This option is useful in combination with prototypes to make C a strongly ''typed'' language.

-Xstack-probe
-X10 Stack checking (probing) is done on some machines. This can be avoided by giving **-X10**.

-Xpass-source
-X11 Outputs the C source as comments in the generated assembly language code.

-Xsigned-bitfields
-X12=0 Bitfields without the **signed** or **unsigned** keyword are treated as signed integers.

-Xunsigned-bitfields
-X12 (Default) Bitfields without the **signed** or **unsigned** keyword are treated as unsigned integers.

-Xswap-cr-nl
-X13 Swaps **\n** and **\r**. Used on systems where carriage return and linefeed are reversed.

-Xsuppress-warnings
-X14 Same as the **-w** switch. No warnings are generated.

-Xunroll=*m*
-X15=*m*
Unrolls small loops this number of times. This is set to 2 as a default if **-O** is given. The number *m* must be a power of 2.

-Xunroll-size=*m*
-X16=*m*

Specifies the number of nodes a loop can contain at the most, to be considered for loop unrolling. Every operator and every operand count as one node, thus the expression **a=b-c;** contains five nodes. The number *m* is set to 20 as a default if **-O** is given.

-Xstruct-best-align
-X17 If this option is given, members of a structure are aligned on their natural alignment instead of the default alignment. This switch produces faster but noncompatible code on machines that put integers on 2-byte boundaries to remain compatible with preceding processors.

-Xinline=*m*
-X19=*m*

Inline functions with less than *m* nodes. This option is available from release 2.36 of the compiler.

-Xparse-size=*m*
-X20=*m*

Waits with code generation of functions until *m* KB of internal memory is used. By delaying generation, the compiler can do intrafunction optimizations like inlining and register tracking. This option is set to 500 as a default.

-Xbottom-up-init
-X21 (Default) Both Kernighan & Ritchie and ANSI-C specify clearly that structure and array initializations with missing braces should be parsed top-down, but some C compilers parse these bottom-up instead. For example:

```
struct z { int a, b; };
struct x {
        struct z    z1[2];
        struct z    z2[2];
} x = { {1,2},{3,4} };

/* could be parsed as
 *      { { {1,2},{0,0} } , { {3,4},{0,0} } }; ansi & k&r
 * or   { { {1,2},{3,4} } , { {0,0},{0,0} } }; bottom-up
 */
```

-Xbottom-up-init makes the compiler parse the above example bottom-up. This option is set when **-Xpcc** (**-X7=3**) is set.

-Xtruncate=*m*
-X22=*m*
Truncates all identifiers after *m* characters.

-Xptr-values-in-a0
-X32 On 68XXX machines that return pointer function values both in register **d0** and **a0**, this option forces the compiler to use the **a0** value instead of the **d0** value. This option is useful when mixing assembler written routines that only return values in **a0**, with C routines.

-Xno-libc-inlining
-X33 Sometimes some **libc** routines like **strcpy**() can be inlined by the compiler. This option prevents this.

-X34=0 The MC68020 with the MC68881/2 floating point co-processor used the **fintrz** instruction to truncate floating-point values before converting them to a 32-bit integer. This causes a problem on the MC68040, because that instruction is not implemented in hardware; rather, it is emulated in the kernel.

-X34=1 (Default) This switch makes the compiler set the floating-point rounding- mode to "round-to-zero" before every float-to-integer conversion and resetting it afterwards. This is done to avoid the kernel-trap that the **fintrz** instruction causes on the MC68040.

-X34=2 This switch uses neither of the above methods to round a floating-point value. It is considerably faster than these constructs, but is dependent on rounding-mode in the processor being set to "round-to-zero" in advance. Note that this mode produces illegal programs if the rounding-mode is not set correctly.

-X35=0 (Default) A 16-bit relative jump-table is to be used for switch-statements. This can cause problems in huge switch-statements (> 32K). The code will be smaller but somewhat slower.

-X35=1 A 32-bit absolute jump-table is to be used for switch-statements. The code is bigger but runs faster.

The **-X7** switch (**-Xk-and-r**, and **-Xpcc**) controls the way the compiler behaves in certain areas. Table 1 explains the cases where this switch has an effect.

Table 1

Case	K-&-R	PCC
long float is same as **double**.	y	y
The **asm** keyword is defined.	y	y
The **volatile**, **const**, and **signed** keywords are defined.	y	n
The type of the hexadecimal constant 0xffffffff is unsigned int (u) or int (i).	i	i
Declarations inside a function using the **extern** keyword have block (b) or file (f) scope. File scope is the same as if the declaration was made outside of the function.	f	f
In ANSI-C, it is legal to initialize automatic arrays and structures/ unions. The compiler always accepts this and is either silent (s) or gives a warning (w).	s	w
When two integers are mixed in an expression, they cause conversions and the result type is either "unsigned wins" (u) or "smallest possible wins" (s). For example, the expression, **((unsigned char)1 > -1)** is 0 if (u) and 1 if (s).	u	u
When prototypes are used and the arguments do not match, the compiler will generate an error (e) or warning (w).	w	w

Table 1 (Continued)

Case	K-&-R	PCC
Float expressions are computed in float (f) or double (d).	d	d
Not fully braced structure and array initializers can either be parsed top-down (t) or bottom-up (b). This is the same as the **-Xbottom-up** (**-X21**) option.	t	b
When pointers and integers are mismatched, the compiler will generate an error (e) or a warning (w). Same as the **-Xmismatch-warning**.	e	w
When an array is declared illegally without a dimension, the compiler will give an error (e) or warning (w).	e	w
Trigraphs, for example, ?? sequences, are either recognized (r) or not (n).	r	n
Illegal structure references give either an error (e) or a warning (w): **int *p; p->a = 1;**	e	w
Comments are replaced by nothing (n) or a space (s).	n	n
Macro arguments are replaced in strings and in character constants: **#define x(a) if (a) printf("a\n");**	y	y

Table 1 (Continued)

Case	K-&-R	PCC
Preprocessor errors are either errors (e) or warnings (w).	e	w
__STDC__ macro is predefined.	n	n
Spaces are legal before cpp #directives.	n	n
Predefined macros like **unix**, **m68k**, and so on, are available.	y	y

The *cc* command also recognizes **-C**, **-D**, **-H**, **-I**, and **-U** and passes these options and their arguments directly to the preprocessor without using the **-W** option. Similarly, the *cc* command recognizes **-l**, **-m**, **-o**, **-r**, **-s**, **-t**, **-u**, **-w**, **-x**, **-z**, **-F**, **-G**, **-L**, **-M**, **-N**, **-V**, and **-Z** and passes these options and their arguments directly to the loader. See the man pages for *cpp*(1) and *ld*(1) for descriptions.

Other arguments are taken to be C-compatible object programs, typically produced by an earlier *cc* run or perhaps libraries of C-compatible routines, and are passed directly to the link editor. These programs, together with the results of any compilations specified, are link edited (in the order given) to produce an executable program with name **a.out** unless the **-o** option of the link editor is used.

The C compiler uses one of four code generators for the 68010, 68020, 68020/68881, and 68040. If 68010 is selected, *cc* invokes *occ* and passes all options directly to *occ*. There are several ways to select a particular code generator, but the selection is normally done using one of two basic mechanisms.

The first is to specify the processor on the *cc* command line, for example, by using the **-Y** option. (An equivalent mechanism is provided by the *gencc*(1M) command, and also by the *cc1sw*(1), *cc2sw*, or *cc2fp* command.) The **-Y** option has additional arguments that allow you to specify pathnames of default libraries, include files, and tools as described earlier.

The second mechanism is to use the CENVIRON shell variable. Note that the first mechanism, specifying the processor and/or search path of libraries and include files, overrides the CENVIRON and any other shell variable settings.

The CENVIRON variable has the following syntax:

CPU=xxxxx,FPU=yyyyy

where CPU indicates the central processor for which code is to be generated and FPU indicates the style of floating-point math to use. Use the following table to determine which combinations are legal and which style of floating-point math will be used.

CPU	FPU		
	SOFTWARE	68881	FPU = omitted
68000	software	illegal	software
68010	software	illegal	software
68020	software	hardware	software
68040	illegal	illegal	hardware

If you are compiling for a 68040-based system but you want to use software floating point, use CPU=68020 and FPU=SOFTWARE, or just CPU=68020.

The FPU parameter may be omitted; the default is either SOFTWARE or HARDWARE, as shown in the table above. The CENVIRON variable should always be set to the appropriate values in the **.profile** or **.cshrc** files or in the makefile. [See *hinv*(1M).]

The C compiler interprets two shell variables which, along with the CENVIRON variable, allow cross-compilation for any CTIX machine:

LIBROOT This variable is a path that is prepended to normal library names when searching for a library. See also *ld*(1).

INCROOT This variable is a path that is prepended to the **/usr/include** and **/usr/include/sys** directories during include file searches. See also *cpp*(1).

The C language standard was extended to allow arbitrary length variable names. The option pair "**-Wp,-T -W0,-XT**" will cause *cc* to truncate arbitrary length variable names to eight characters.

FILES

file.c	C source file
file.i	preprocessed C source file
file.o	object file

file.s	assembly language file
a.out	link edited output
LIBDIR/*crt1.o	start-up routine
LIBDIR/crtn.o	start-up routine
TMPDIR/*	temporary files
BINDIR/as	assembler, *as*(1)
BINDIR/ld	link editor, *ld*(1)
LIBDIR/libc.a	standard C library
LIBDIR/libc_s.a	standard C shared library

LIBDIR is usually /lib
BINDIR is usually /bin
TMPDIR is usually /tmp but can be redefined by setting the environment variable **TMPDIR** [see *tempnam*( ) in *tmpnam*(3S)].

SEE ALSO

as(1), ld(1), cpp(1), gencc(1), lint(1), prof(1), sdb(1), tmpnam(3S).
Kernighan, B. W., and Ritchie, D. M., *The C Programming Language*, Prentice-Hall, 1978.
Programmer's Guide: CTIX Supplement

CAVEATS

cc will complain if it encounters inconsistencies between the processor selected and default libraries or include files. *occ* is invoked to generate code for 68000 or 68010 processors.

Sometimes the range for a branch does not fit inside a word; in this case, an error message is printed. For suggested workarounds, see the section called "Span-Dependent Optimization" in Chapter 14 of the *Programmer's Guide: CTIX Supplement.*

DIAGNOSTICS

The diagnostics produced by the C compiler are sometimes cryptic. Occasional messages may be produced by the assembler or link editor.

NOTES

By default, the return value from a compiled C program is completely random. The only two guaranteed ways to return a specific value is to explicitly call *exit*(2) or to leave the function **main**( ) with a "*return expression;*" construct.

NAME

cc1sw, cc2sw, cc2fp, cc4 - front-end to the cc command

SYNOPSIS

cc1sw [options] files
cc2sw [options] files
cc2fp [options] files
cc4 [options] files

DESCRIPTION

cc1sw, *cc2sw*, *cc2fp*, and *cc4* provide a front-end to *cc* for use in cross-compilation. *cc1sw* generates code for a 68010 processor with software floating point, *cc2sw* generates code for a 68020 processor with software floating point, *cc2fp* generates code for a 68020 processor with hardware floating point, and *cc4* generates code for a 68040 processor. The commands call *cc* with the following **-Y** options:

cc1sw **-Y c,68010**
-Y S,/cross/1sw/lib
-Y L,/cross/1sw/lib
-Y U,/cross/1sw/usr/lib

cc2sw **-Y c,68020**
-Y S,/cross/2sw/lib
-Y L,/cross/2sw/lib
-Y U,/cross/2sw/usr/lib

cc2fp **-Y c,68881**
-Y S,/cross/2fp/lib
-Y L,/cross/2fp/lib
-Y U,/cross/2fp/usr/lib

cc4 **-Y c,68040**
-Y S,/cross/2fp/lib
-Y L,/cross/2fp/lib
-Y U,/cross/2fp/usr/lib

Options are those options available for *cc*.

The default include directories searched by *cc* (called by *cc1sw*, *cc2sw*, or *cc2fp*) are **/usr/include** and **/usr/include/sys**. The default include directories can be overridden by using the **-Y I,***dirname* option or setting the INCROOT environment variable [see *cc*(1)].

FILES

file.c	C source file
file.i	preprocessed C source file
file.o	object file
file.s	assembly language file
a.out	link edited output
LIBDIR/*crt1.o	start-up routine
LIBDIR/crtn.o	start-up routine
TMPDIR/*	temporary files
LIBDIR/cpp	preprocessor, *cpp*(1)
BINDIR/as	assembler, *as*(1)
BINDIR/ld	link editor, *ld*(1)
LIBDIR/	standard C library
LIBDIR/libc_s.a	standard C shared library

LIBDIR is usually /lib
BINDIR is usually /bin
TMPDIR is usually /tmp but can be redefined by setting the environment variable **TMPDIR** [see *tempnam*( ) in *tmpnam*(3S)].

NOTES

cc2sw can be used for generating software floating point on 68040-based systems.

SEE ALSO

cc(1), gencc(1M).

NAME

config - configure a CTIX system

SYNOPSIS

/etc/config [**-l** file] [**-c** file] [**-m** file] [**-t**] [**-b** num] [**-d** num] [**-s** num] [**-f** num] dfile

DESCRIPTION

The *config* program takes a description of a CTIX system, generates a configuration table file, and generates a hardware interface file. The configuration table file is a C program defining the configuration tables for the various devices on the system. The hardware interface file provides information about the interface between the hardware and device handlers.

The options to *config* (1M) are as follows:

-l Specifies the name of the hardware interface file; **low.s** is the default.

-c Specifies the name of the configuration table file; **conf.c** is the default.

-m Specifies the name of the file that contains information about supported devices; **/etc/master** is the default name. This file is supplied with the CTIX system and should *not* be modified unless the user *fully* understands its construction.

-t Requests a short table of major device numbers for character- and block-type devices. This can facilitate the creation of special files.

-b Specifies the minimum number of entries in the *bdevsw* array. The default value is **20**.

-d Specifies the minimum number of entries in the *cdevsw* array. The default value is **128**.

-s Specifies the minimum number of entries in the *fmodsw* array. The default value is **16**.

-f Specifies the minimum number of entries in the *fstypsw* array. The default value is **8**. The **-b**, **-d**, **-s**, and **-f** options are provided to ensure that a sufficient number of empty slots are available for loadable modules, such as drivers, stream modules, software modules, and file system types.

The user must supply *dfile*, which must contain device information for the user's system. The *dfile* is divided into two parts: the first part contains physical device specifications; the second part contains system-dependent information. Any line with an asterisk (*) in column 1 is a comment. A sample *dfile* file is provided in the **/usr/sys/cf** directory.

First Part of dfile

Each line in the first part of the *dfile* contains one field, *devname*, which is the name of the device, software module, stream module, or file system type (as it appears in the **/etc/master** device table).

The disk driver section is the first group of the first part; this first group must contain only disk driver *devname* entries.

Note that for disk controllers, the position of *devname* in *dfile* determines the **/dev/dsk** controller number assigned to the *devname* driver. For tape controllers, the position of *devname* has no significance. The tape controller number mapping is performed by the CTIX installation tools, using the **mknod**(1M) command.

The following example shows how controller numbers are assigned from an S/640 or S/480 *dfile*:

dfile Entry	Controller Number
diskonbd	**/dev/dsk/c0d? (ST506)**
Vsmd3200	**/dev/dsk/c1d? (1st SMD)**
scsidisk	**/dev/dsk/c2d? (SCSI)**
Vsmd3200	**/dev/dsk/c3d? (2nd SMD)**
:	
scsi	**(required line for SCSI)**
stape	**(SCSI QIT and HIT)**

In the example, above the quarter-inch tape (QIT) drive is a SCSI device, assigned to **/dev/rmt/c0d***?*. The VME-based half-inch tape controller is assigned **/dev/rmt/c1d***?* by the installation tools. The driver (**/etc/lddrv/ipt.o**) is loaded dynamically at boot time, so **ipt** is not entered in the *dfile*.

For the S/120, S/22*x*, and S/320, disk controller numbers are assigned as they are on the S/640 and S/480, but tape controller numbers are assigned differently, as shown below:

dfile Entry	Controller Number
:	
qic	**/dev/rmt/c0d0 (QIC-2)**
scsi	**(required line for SCSI)**
stape	**(SCSI QIT and HIT)**

As shown above, the first quarter-inch tape drive is QIC-2 controller-based; it is assigned **/dev/rmt/c0d0**. Again, the VME-based half-inch tape drive controller

is assigned **/dev/rmt/c1d**? by the installation tools. The SCSI quarter-inch tape and half-inch tape drives are assigned **/dev/rmt/c0d1** to **/dev/rmt/c0d7**.

For the S/80 and S/280, the controller number assignment is different, because the onboard disk controller is the SCSI controller and there is no VME expansion:

dfile Entry	Controller Number
scsidisk	**/dev/dsk/c0d? (SCSI)**
:	
scsi	**(required line for SCSI)**
stape	**(SCSI QIT and HIT)**

As shown above, the first quarter-inch tape drive is assigned **/dev/rmt/c0d0**. Remaining tape drives are assigned **/dev/rmt/c0d**? (where ? is 1 through 7).

Second Part of dfile

The second part of the *dfile* contains four types of lines, listed and described below. Note that *all* specifications of this part *are required*, although the order is arbitrary.

1. *Root/pipe device specification*

   Two lines of three fields each:

   root devname minor
   pipe devname minor

   where *minor* is the minor device number (in decimal) of the slice on the fixed disk.

2. *Swap device specification*

   One line that contains five fields, as follows:

   swap devname minor swplo nswap

   where *swplo* is the lowest disk block (decimal) in the swap area and *nswap* is the maximum number of 1-KB disk blocks (decimal) in the swap area. The kernel sizes the actual swap area size and configures itself for up to this maximum.

3. *Dynamic device number assignment*

   The *devnames* for **root**, **swap**, and **pipe** can be specified as **any**. The major device numbers are undetermined until boot time.

The key word parameter **dynamic** can be used with the **any** *devname* to force the major device number of the **boot** device to 0. For example, if the **devnames** in the *dfile* are **diskonbd** and **scsidisk**, and the **boot** device is the **scsidisk**, the kernel swaps the device numbers so that **scsidisk** can be accessed through major device number 0 instead of 1, as specified in the *dfile*. All access to SCSI disks is through **/dev/dsk/c0d**?**s**?, and all access to ST506 disks is through **/dev/dsk/c1d**?**s**?. If **any** is not used, **dynamic** has no effect.

4. *Parameter specification*

Any number of lines of two fields each, chosen from the following list. *Number* is decimal. Note that the following parameter list is *not* complete; parameters not on the list either must not be changed or have no effect.

```
buffers      number   /* number of 1024-byte file system
                         caching buffers */
buffers_4k number     /* number of 4096-byte file system
                         caching buffers */
dmmxsz       number   /* max number of pages per loadable
                         driver */
inodes       number   /* max open inodes in system */
s5inodes     number   /* max open s5inodes in system */
files        number   /* max open files in system */
flkrec       number   /* max locks active in system */
mounts       number   /* max file systems mounted */
regions      number   /* total number of regions in system */
procs        number   /* max processes in system */
maxproc      number   /* max processes per user ID */
maxfsiz      number   /* ulimit default in 512-byte blocks */
maxumem      number   /* max number of pages per process */
cbufsize     number   /* console circular buffer size in bytes */
msgmax       number   /* max chars in a message */
msgmni       number   /* max active message queues */
msgmnb       number   /* max total chars in message queues */
msgtql       number   /* max messages in system */
msgssz       number
msgseg       number   /* msgssz * msgseg = number bytes of
                         system buffering */
nlldrv       number   /* max number of loadable drivers */
semmni       number   /* max active semaphores */
semmns       number   /* max semaphores in system */
semmsi       number   /* max semaphores per ID */
```

```
semopm      number   /* max operations per semop call */
semume      number   /* max undo structures per process */
semmnu      number   /* max undo structures in system */
diriosz     number   /* direct I/O default size */
shmmax      number   /* max bytes in a shared segment */
shmmin      number   /* min bytes in a shared segment */
shmmni      number   /* max active shared segments */
shmseg      number   /* max attached segments per process */
shmbrk      number   /* gap in pages between data and
                        shared memory */
nqueue      number   /* max stream queues */
nstream     number   /* max streams */
nblk4096    number   /* number of 4096 byte stream bufs */
nblk2048    number   /* number of 2048 byte stream bufs */
nblk1024    number   /* number of 1024 byte stream bufs */
nblk512     number   /* number of 512 byte stream bufs */
nblk256     number   /* number of 256 byte stream bufs */
nblk128     number   /* number of 128 byte stream bufs */
nblk64      number   /* number of 64 byte stream bufs */
nblk16      number   /* number of 16 byte stream bufs */
nblk4       number   /* number of 4 byte stream bufs */
shlbmax     number   /* max # of shared libs per process */
nofiles     number   /* max # of open files per process */
ntimod      number   /* max # of TLI connections */
ntirdwr     number   /* max # of TLI read/write connections */
nsp         number   /* max # of stream pipes */
```

Certain parameters, if set to 0, allow the kernel to autoconfigure. For example, **procs**, **regions**, **clists**, **i-nodes**, **s5inodes**, **files**, and **buffers** are autoconfigurable. The value of **procs** is based on the number of users; the values for **regions**, **i_nodes**, **s5inodes**, and **files** are based on the value of **procs**. The value of **clists** is based on the number of serial and cluster ports. The value of **buffers** is based on the amount of physical memory. Any or all of the autoconfigured values can be overridden. The value of **maxumem** can also be set to 0; in which case, it floats between 1 MB and one quarter of the total swap space.

EXAMPLE

This example assumes an S/640 system with the following devices:

- onboard ST506 disks (*root*)
- first interphase SMD disk controller

- SCSI disks
- second interphase SMD disk controller
- RS-232-C (any number of ports)
- SCSI tape drives
- one parallel line printer
- root device is a disk (drive 0, section 1)
- pipe device is a disk (drive 0, section 1)
- swap device is a disk (drive 0, section 2), with a swplo of 1 and an nswap of 8000
- number of buffers is 100
- number of processes is 100
- maximum number of processes per user ID is 25
- number of mounts is 6
- number of inodes is 100
- number of files is 120
- number of character buffers is 64
- messages are to be included
- semaphores are to be included

The S/640 system configuration would be specified as follows in the *dfile*:

```
diskonbd
Vsmd3200
scsidisk
Vsmd3200
serial
scsi
stape
console
plp
root    any             01
pipe    any             01
swap    any             02      0       8000
* Comments are inserted in this manner
buffers 100
procs   100
maxproc 25
mounts  6
inodes  100
files   120
mesg    1
sema    1
clists  64
```

FILES

/etc/master	default input master device table
/usr/sys/cf/dfile	default system configuration
/usr/sys/cf/low.s	default output hardware interface
/usr/sys/cf/conf.c	default output configuration table

SEE ALSO

ldeeprom(1M), uconf(1M), master(4).
S/Series CTIX Administrator's Guide.

DIAGNOSTICS

Diagnostics are routed to the standard output and are self-explanatory.

BUGS

The **-t** option does not know about devices that have aliases.

NAME

cpp - the C language preprocessor

SYNOPSIS

LIBDIR/cpp [option ...] [ifile [ofile]]

DESCRIPTION

The C language preprocessor, *cpp*, is invoked as the first pass of any C compilation by the *cc*(1) command. Thus, *cpp*'s output is designed to be in a form acceptable as input to the next pass of the C compiler. If you are running a 6.3 or later release of CTIX, *cpp* is a built-in feature of the C compiler. See *m4*(1) for a general macro processor. The use of *cpp* other than through the *cc*(1) command is not suggested, since the functionality of *cpp* has been moved in the latest release.

cpp optionally accepts two filenames as arguments. *Ifile* and *ofile*, respectively, are the input and output for the preprocessor. They default to standard input and standard output if not supplied.

The following *options* to *cpp* are recognized:

-P Preprocesses the input without producing the line control information used by the next pass of the C compiler.

-C By default, *cpp* strips C-style comments. If the **-C** option is specified, all comments (except those found on *cpp* directive lines) are passed along.

-U*name* Removes any initial definition of *name*, where *name* is a reserved symbol that is predefined by the particular preprocessor. Following is the current list of these possibly reserved symbols.

operating system:	unix, gcos, ibm, os, tss
hardware:	interdata, mc68k, mc68k32, m68k, mc68000, mc68010, mc68020, mc68881, mc68040, pdp11, u370, u3b, u3b5, u3b2, u3b20d, vax
UNIX system variant:	RES, RT
lint(1):	lint

-D*name*
-D*name=def*

Defines *name* with value *def* as if by a **#define**. If no *=def* is given, *name* is defined with value 1. The **-D** option has lower precedence than the **-U** option. That is, if the same name is used in both a

-U option and a **-D** option, the name will be undefined regardless of the order of the options.

-T The **-T** option forces *cpp* to use only the first eight characters to distinguish preprocessor symbols and is included for backward compatibility.

-I*dir* Changes the algorithm for searching for **#include** files whose names do not begin with / to look in *dir* before looking in the directories on the standard list. Thus, **#include** files whose names are enclosed in " " will be searched for first in the directory of the file with the **#include** line, then in directories named in **-I** options, and last in directories on a standard list. For **#include** files whose names are enclosed in <>, the directory of the file with the **#include** line is not searched. By default, *cpp* searches for the name enclosed in < > in **/usr/include**; however, if the shell variable INCROOT is set, *cpp* prepends the value of INCROOT to the standard list. This is particularly useful for cross-compilation.

-Y*dir* Uses directory *dir* in place of the standard list of directories when searching for **#include** files. The **-Y** option overrides the value for INCROOT if it is set.

-H Prints, one per line on standard error, the pathnames of included files.

Two special names are understood by *cpp*. The name **__LINE__** is defined as the current line number (as a decimal integer) as known by *cpp*, and **__FILE__** is defined as the current filename (as a C string) as known by *cpp*. They can be used anywhere (including in macros) just as any other defined name.

All *cpp* directive lines start with # in column 1. Any number of blanks and tabs is allowed between the # and the directive. The directives are as follows:

#define *name token-string*

Replaces subsequent instances of *name* with *token-string*.

#define *name*(*arg*, ..., *arg*) *token-string*

Notice that there can be no space between *name* and the (. Replace subsequent instances of *name* followed by a (, a list of comma-separated sets of tokens, and a) followed by *token-string*, where each occurrence of an *arg* in the *token-string* is replaced by the corresponding set of tokens in the comma-separated list. When a macro with arguments is expanded, the arguments are placed into the expanded *token-string* unchanged. After the entire *token-string* has been expanded, *cpp* restarts its scan for names to expand at the beginning of the newly created *token-string*.

#undef *name*

Causes the definition of *name* (if any) to be forgotten from now on. No additional tokens are permitted on the directive line after *name*.

#ident *"string"*

Puts *string* into the .comment section of an object file.

#include *"filename"*
#include *<filename>*

Includes at this point the contents of *filename* (which will then be run through *cpp*). When the *<filename>* notation is used, *filename* is only searched for in the standard places. See the **-I** and **-Y** options above for more detail. No additional tokens are permitted on the directive line after the final " or >.

#line *integer-constant "filename"*

Causes *cpp* to generate line control information for the next pass of the C compiler. *Integer-constant* is the line number of the next line and *filename* is the file from which it comes. If *"filename"* is not given, the current filename is unchanged. No additional tokens are permitted on the directive line after the optional *filename*.

#endif

Ends a section of lines begun by a test directive (**#if**, **#ifdef**, or **#ifndef**). Each test directive must have a matching **#endif**. No additional tokens are permitted on the directive line.

#ifdef *name*

The lines following will appear in the output if and only if *name* has been the subject of a previous **#define** without being the subject of an intervening **#undef**. No additional tokens are permitted on the directive line after *name*.

#ifndef *name*

The lines following will appear in the output if and only if *name* has not been the subject of a previous **#define**. No additional tokens are permitted on the directive line after *name*.

#if *constant-expression*

Lines following will appear in the output if and only if the *constant-expression* evaluates to nonzero. All binary nonassignment C operators, the **?:** operator, the unary –, **!**, and ~ operators are all legal in *constant-expression*. The precedence of the operators is the same as defined by the C language. There is also a unary operator **defined**, which can be used in *constant-expression* in these two forms: **defined**

(*name*) or **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 *constant-expression*. In particular, the **sizeof** operator is not available.

To test whether either of two symbols, *foo* and *fum*, are defined, use

```
#if defined(foo) || defined(fum)
```

#elif *constant-expression*

An arbitrary number of **#elif** directives is allowed between a **#if**, **#ifdef**, or **#ifndef** directive and a **#else** or **#endif** directive. The lines following the **#elif** directive will appear in the output if and only if the preceding test directive evaluates to zero, all intervening **#elif** directives evaluate to zero, and the *constant-expression* evaluates to nonzero. If *constant-expression* evaluates to nonzero, all succeeding **#elif** and **#else** directives will be ignored. Any *constant-expression* allowed in a **#if** directive is allowed in a **#elif** directive.

#else The lines following will appear in the output if and only if the preceding test directive evaluates to zero, and all intervening **#elif** directives evaluate to zero. No additional tokens are permitted on the directive line.

The test directives and the possible **#else** directives can be nested.

FILES

INCDIR	standard directory list for **#include** files, usually /usr/include
LIBDIR	usually /lib

SEE ALSO

cc(1), lint(1), m4(1).

DIAGNOSTICS

The error messages produced by *cpp* are intended to be self-explanatory. The line number and filename where the error occurred are printed along with the diagnostic.

NOTES

The unsupported **-W** option enables the **#class** directive. If it encounters a **#class** directive, *cpp* will exit with code 27 after finishing all other processing. This option provides support for "C with classes."

Because the standard directory for included files may be different in different environments, this form of **#include** directive:

```
#include <file.h>
```

should be used, rather than one with an absolute path, like:

```
#include "/usr/include/file.h"
```

cpp warns about the use of the absolute pathname.

NAME

crash - examine system images

SYNOPSIS

/etc/crash [**-d** dumpfile] [**-n** namelist] [**-w** outputfile]

DESCRIPTION

The *crash* command is used to examine the system memory image of a live or a crashed system by formatting and printing control structures, tables, and other information. Command line arguments to *crash* are *dumpfile*, *namelist*, and *outputfile*.

dumpfile is the file containing the system memory image. The default *dumpfile* is **/dev/kmem**. The system image can also be slice zero of the raw disk that contains the dump area (for example, **/dev/rdsk/c0d0s0**); or it can be the pathname of a file produced using *dd* to copy slice zero or just the dump area; or in the case of a tape dump, the second file on the tape.

The unstripped executable file *namelist* contains the symbol table information needed for symbolic access to the system memory image to be examined. The default *namelist* is **/etc/lddrv/unix.exec** if examining a running system or **/etc/lddrv/prev.unix.exec** if examining a dump. If neither of these files exists, the default is **/unix**. If a system image from another machine is to be examined, the corresponding **prev.unix.exec** must be copied from that machine. The **prev.unix.exec** is preferred to **/unix** because it also contains the *namelist* for all the loaded drivers at the correct addresses.

When the *crash* command is invoked, a session is initiated. The output from a *crash* session is directed to *outputfile*. The default *outputfile* is the standard output.

Input during a *crash* session is of the form:

function [argument ...]

where *function* is one of the *crash* functions described in the FUNCTIONS section of this man page, and *arguments* are qualifying data that indicate which items of the system image are to be printed.

The default for process-related items is the current process for a running system and the process that was running at the time of the crash for a crashed system. If the contents of a table are being dumped, the default is all active table entries.

The following function options are available to *crash* functions wherever they are semantically valid:

-e	Displays every entry in a table.
-f	Displays the full structure.
-p	Interprets all address arguments in the command line as *physical* addresses.
-s process	Specifies a process slot other than the default.
-w file	Redirects the output of a function to *file*.

Note that if the **-p** option is used, all address and symbol arguments explicitly entered on the command line will be interpreted as physical addresses. If they are not physical addresses, results will be inconsistent.

The functions *mode*, *defproc*, and *redirect* correspond to the function options **-p**, **-s**, and **-w**. The *mode* function may be used to set the address translation mode to physical or virtual for all subsequently entered functions; *defproc* sets the value of the process slot argument for subsequent functions; and *redirect* redirects all subsequent output.

Output from *crash* functions may be piped to another program in the following way:

function [argument ...] ! shell_command

For example,

```
mount ! grep rw
```

will write all mount table entries with an *rw* flag to the standard output. The redirection option (**-w**) cannot be used with this feature.

Depending on the context of the function, numeric arguments will be assumed to be in a specific radix. Counts are assumed to be decimal. Addresses are always hexadecimal. Table address arguments larger than the size of the function table will be interpreted as hexadecimal addresses; those smaller will be assumed to be decimal slots in the table. Default bases on all arguments may be overridden. The C conventions for designating the bases of numbers are recognized. A number that is usually interpreted as decimal will be interpreted as hexadecimal if it is preceded by *0x* and as octal if it is preceded by *0*. Decimal override is designated by *0d*, and binary by *0b*.

Aliases for functions may be any uniquely identifiable initial substring of the function name. Traditional aliases of one letter, such as *p* for *proc*, remain valid.

Many functions accept different forms of entry for the same argument. Requests for table information will accept a table entry number, a physical address, a virtual address, a symbol, a range, or an expression. A range of slot numbers may be specified in the form *a-b* where *a* and *b* are decimal numbers. An expression consists of two operands and an operator. An operand may be an address, a symbol, or a number; the operator may be +, -, *, /, &, or |. An operand that is a number should be preceded by a radix prefix if it is not a decimal number (*0* for octal, *0x* for hexadecimal, and *0b* for binary). The expression must be enclosed in parentheses ( ). Other functions will accept any of these argument forms that are meaningful.

Two abbreviated arguments to *crash* functions are used throughout. Both accept data entered in several forms. They may be expanded into the following:

table_entry = table entry | address | symbol | range | expression

start_addr = address | symbol | expression

FUNCTIONS

? [**-w** file]
List available functions.

!cmd Escapes to the shell to execute a command.

adv [**-e**] [**-w** file] [[**-p**] table_entry ...]
Prints the advertise table.

base [**-w** file] number ...
Prints *number* in binary, octal, decimal, and hexadecimal. A number in a radix other then decimal should be preceded by a prefix that indicates its radix as follows: *0x*, hexidecimal; *0*, octal; and *0b*, binary.

buffer [**-w** file] [-format] bufferslot

or

buffer [**-w** file] [-format] [**-p**] start_addr
Alias: **b**.
Prints the contents of a buffer in the designated format. The following format designations are recognized: **-b**, byte; **-c**, character; **-d**, decimal; **-x**, hexadecimal; **-o**, octal; **-r**, directory; and **-i**, inode. If no format is given, the previous format is used. The default format at the beginning of a *crash* session is hexadecimal.

bufhdr [**-f**] [**-w** file] [[**-p**] table_entry ...]
Alias: **buf**.
Prints system buffer headers.
The **-f** option produces different output depending on whether the buffer is local or remote (contains RFS data).

callout [**-w** file]
Alias: **c**.
Prints the callout table.

cblk [**-e**] [**-p**] [**-w** file] [**-t** type] [table_entry ...]
Displays contents of cblocks.

cd Equivalent to

tty -t cd

(See **tty** function below.)

clist [**-e**] [**-p**] [**-w** file] [**-t** type] [table_entry ...]
Displays usage of clists.

conbuf [**-w** file]
Displays console buffer.

dballoc [**-w** file] [class ...]
Prints the dballoc table. If a class is entered, only data block allocation information for that class will be printed.

dbfree [**-w** file] [class ...]
Prints free streams data block headers. If a class is entered, only data block headers for the class specified will be printed.

dblock [**-e**] [**-w** file] [**-c** class ...]

or

dblock [**-e**] [**-w** file] [[**-p**] table_entry ...]
Prints allocated streams data block headers. If the class option (**-c**) is used, only data block headers for the class specified will be printed.

defproc [**-w** file] [**-c**]

or

defproc [**-w** file] [slot]
Sets the value of the process slot argument. The process slot argument may be set to the current slot number (**-c**) or the slot number may be specified. If no argument is entered, the value of the previously set slot number is printed. At the start of a *crash* session, the process slot is set to the current process.

dis [**-w** file] [**-a**] start_addr [count]
Disassembles from the start address for *count* instructions. The default count is 1. The absolute option (**-a**) specifies a nonsymbolic disassembly.

disk [**-w** file]
Displays disk information.

drvtable [**-w** file]
Displays loadable driver table information.

ds [**-w** file] virtual_address ...
Prints the data symbol whose address is closest to, but not greater than, the address entered.

fcallout [**-w** file]
Alias: **fc**.
Prints the fast callout table.

file [**-e**] [**-w** file] [[**-p**] table_entry ...]
Alias: **f**.
Prints the file table.

findaddr [**-w** file] table slot
Prints the address of *slot* in *table*. Only tables available to the *size* function are available to *findaddr*.

findslot [**-w** file] virtual_address ...
Prints the table, entry slot number, and offset for the address entered. Only tables available to the *size* function are available to *findslot*.

fs [**-w** file] [[**-p**] table_entry ...]
Prints the file system information table.

gdp [**-e**] [**-f**] [**-w** file] [[**-p**] table_entry ...]
Prints the gift descriptor protocol table.

gt Equivalent to

tty -t gt

(See **tty** function below.)

help [**-w** file] function ...
Prints a description of the named function, including syntax and aliases.

inode [**-e**] [**-f**] [**-w** file] [[**-p**] table_entry ...]
Alias: **i**.
Prints the inode table, including file system switch information.

iop16 Equivalent to

tty -t iop16

(See **tty** function below.)

kfp [**-w** file] [**-s** process] [**-r**]

or

kfp [**-w** file] [**-s** process] [value]
Prints the frame pointer for the start of a kernel stack trace. The kfp value can be set using the value argument or the reset option (**-r**), which sets the kfp from the saved kfp in a dump. If no argument is entered, the current value of the kfp is printed.

lck [**-e**] [**-w** file] [[**-p**] table_entry ...]
Alias: **l**.
Prints record locking information. If the **-e** option is used or table address arguments are given, the record lock list is printed. If no argument is entered, information on locks relative to inodes is printed.

linkblk [**-e**] [**-w** file] [[**-p**] table_entry ...]
Prints the linkblk table.

major [**-w** file] [entry ...]
Prints the MAJOR table.

map [**-w** file] mapname ...
Prints the map structure of the given mapname.

mbfree [**-w** file]
Prints free streams message block headers.

mblock [**-e**] [**-w** filename] [[**-p**] table_entry ...]
Prints allocated streams message block headers.

mode [**-w** file] [mode]
Sets address translation of arguments to virtual (**v**) or physical (**p**) mode. If no mode argument is given, the current mode is printed. At the start of a *crash* session, the mode is virtual.

mount [**-e**] [**-w** file] [[**-p**] table_entry ...]
Alias: **m**.
Prints the mount table.

msg [**-e**] [**-f**] [**-p**] [**-w** file] [**-s** process] [table_entry ...]
Displays IPC message queue headers.

msginfo [**-p**] [**-w** file]
Displays IPC message information.

msgtext [**-e**] [**-p**] [**-w** file] [**-s** process] [table_entry ...]
Displays IPC message data.

nm [**-w** file] symbol ...
Prints value and type for the given symbol.

notify [**-e**] [**-p**] [**-w** file] symbols

od [**-p**] [**-w** file] [**-**format] [-mode] [**-s** process] start_addr [count]
Alias: **rd**.
Prints *count* values starting at the start address in one of the following formats: character (**-c**), decimal (**-d**), hexadecimal (**-x**), octal (**-o**), ASCII (**-a**), or hexadecimal/character (**-h**), and one of the following modes: long (**-l**), short (**-t**), or byte (**-b**). The default mode for character and ASCII formats is byte; the default mode for decimal, hexadecimal, and octal formats is long. The format **-h** prints both hexadecimal and character representations of the addresses dumped; no mode needs to be specified. When format or mode is omitted, the previous value is used. At the start of a *crash* session, the format is hexadecimal and the mode is long. If no count is entered, 1 is assumed.

pdt [**-e**] [**-w** file] [**-s** process] section segment

or

pdt [**-e**] [**-w** file] [**-s** process] [**-p**] start_addr [count]
S/640 Only:
The page descriptor table of the designated memory *section* and *segment* is printed. Alternatively, the page descriptor table starting at the start address for *count* entries is printed. If no count is entered, 1 is assumed.

pfdat [**-e**] [**-w** file] [[**-p**] table_entry ...]
Prints the pfdata table.

pfree [**-e**] [**-p**] [**-w** file] table_entry ...
Displays free list entries.

phash [**-e**] [**-p**] [**-w** file]
Displays page hash table.

proc [-e] [**-f**] [**-w** file] [[**-p**] table_entry ... #procid ...]

or

proc [**-f**] [**-w** file] [**-r**]
Alias: **p**.
Prints the process table. Process table information may be specified in two ways. First, any mixture of table entries and process ids may be entered. Each process id must be preceded by a #. Alternatively, process table information for runnable processes may be specified with the runnable option (**-r**).

pt Equivalent to

tty -t pt

(See **tty** function below.)

qrun [**-w** file]
Prints the list of scheduled streams queues.

queue [**-e**] [**-w** file] [[**-p**] table_entry ...]
Prints streams queues.

quit Alias: **q**.
Terminates the *crash* session.

rcvd [**-e**] [**-f**] [**-w** file] [[**-p**] table_entry ...]
Prints the receive descriptor table.

redirect [**-w** file] [**-c**]

or

redirect [**-w** file] [file]
Used with a filename, redirects output of a *crash* session to the named file. If no argument is given, the filename to which output is being redirected is printed. Alternatively, the close option (**-c**) closes the previously set file and redirects output to the standard output.

region [**-e**] [**-f**] [**-w** file] [[**-p**] table_entry ...]
Prints the region table.

scsi [**-w** file]
Displays SCSI tables.

scsirqb [**-f**] [**-w** file] [tbl_entry | start_addr]
Displays SCSI request blocks.

sdt [**-e**] [**-w** file] [**-s** process] section

or

sdt [**-e**] [**-w** file] [**-s** process] [**-p**] start_addr [count]
S/640 Only:
The segment descriptor table for the named memory section is printed. Alternatively, the segment descriptor table starting at start address for *count* entries is printed. If no count is given, a count of 1 is assumed.

search [**-p**] [**-w** file] [**-m** mask] [**-s** process] pattern start_addr length
Prints the words in memory that match *pattern*, beginning at the start address for *length* words. The mask is anded (&) with each memory word and the result compared against the pattern. The mask defaults to 0xffffffff.

ser Equivalent to

tty -t ser

(See **tty** function below.)

shm [**-e**] [**-f**] [**-p**] [**-w** file] table_entry ...
Displays IPC shared memory headers.

shminfo [**-p**] [**-w** file]
Displays system IPC shared memory information.

size [**-w** file] [**-x**] [structure_name ...]
Prints the size of the designated structure. The (**-x**) option prints the size in hexadecimal. If no argument is given, a list of the structure names for which sizes are available is printed.

sndd [**-e**] [**-f**] [**-w** file] [[**-p**] table_entry ...]
Prints the send descriptor table.

sptb [**-e**] [**-p**] [**-w** file] [start_addr]
Displays sptballoc maps.

srmount [**-e**] [**-w** file] [[**-p**] table_entry ...]
Prints the server mount table.

stack [**-w** file] [**-u**] [process]

or

stack [**-w** file] [**-k**] [process]

or

stack [**-w** file] [[**-p**] **-i** start_addr]
Alias: **s**.
Dumps stack. The (**-u**) option prints the user stack. The (**-k**) option prints the kernel stack. The (**-i**) option prints the interrupt stack starting at the start address. If no arguments are entered, the kernel stack for the current process is printed. The interrupt stack and the stack for the current process are not available on a running system.

stat [**-w** file]
Prints system statistics.

stream [**-e**] [**-f**] [**-w** file] [[**-p**] table_entry ...]
Prints the streams table.

strstat [**-w** file]
Prints streams statistics.

swap Displays swap map statistics.

swapinfo
Displays swap statistics.

trace [**-w** file] [**-r**] [process]

or

trace [**-w** file] [[**-p**] **-i** start_addr]
Alias: **t**.
Prints stack trace. The kfp value is used with the **-r** option. The interrupt option prints a trace of the interrupt stack beginning at the start address. The interrupt stack trace and the stack trace for the current process are not available on a running system.

ts [**-w** file] virtual_address ...
Prints closest text symbol to the designated address.

tty [**-e**] [**-f**] [**-w** file] [**-t** type [[**-p**] table_entry ...]]

or

tty [**-e**] [**-f**] [**-w** file] [[**-p**] start_addr]
Valid types: **cd**, **gt**, **iop16**, **pt**, **ser**, **vt**, **vtd**, **wxt**.
Prints the tty table. If no arguments are given, the tty table for all tty types is printed. If the **-t** option is used, the table for the single tty type specified is printed. If no argument follows the type option, all entries in the table are printed. A single tty entry may be specified from the start address.

unnotify [**-e**] [**-p**] [**-w** file] [**-s** process] symbols
Displays queued notifications for process.

user [**-f**] [**-w** file] [process]
Alias: **u**.
Prints the ublock for the designated process.

var [**-w** file]
Alias: **v**.
Prints the tunable system parameters.

vt Equivalent to

tty -t vt

(See **tty** function above.)

vtd Equivalent to

tty -t vtd

(See **tty** function above.)

vtop [**-w** file] [**-s** process] start_addr ...
Prints the physical address translation of the virtual start address.

wxt Equivalent to

tty -t wxt

(See **tty** function.)

FILES

/dev/kmem	system image of currently running system
/dev/rdsk/c?d?s0	used to access system image on disk

NAME

createdev - create device nodes for assorted device types

SYNOPSIS

createdev -d device [**-c** controller] [**-v**] [**-r**] [**-p**] [**-t**]

DESCRIPTION

The *createdev* command is used to create device nodes of various types. After parsing various parameters, the command invokes *mknod* to create the specified device node or sets of device nodes.

The **-d** option specifies the device number (for example, unit, drive, or line number), and is required for every invocation.

The **-c** option specifies the controller number of the specified device.

The **-v** option specifies that disk devices are to be created. Both block-type and character-type device nodes of the form **/dev/rdsk/c***x***d***y***s***z** and **/dev/dsk/c***x***d***y***s***z** are added. Each invocation creates as many slices as the disk supports on CTIX (currently 16). For the -v option, both the **-d** and **-c** options are required.

The **-r** option specifies that devices are created to provide access to streaming tape drives; these are the character-type device nodes. These devices are of the form: **/dev/rmt/c***x***d***y*, **/dev/rmt/c***x***d***y***c**, **/dev/rmt/c***x***d***y***h**, **/dev/rmt/c***x***d***y***hn**, **/dev/rmt/c***x***d***y***l**, **/dev/rmt/c***x***d***y***ln**, **/dev/rmt/c***x***d***y***m**, **/dev/rmt/c***x***d***y***mn**, and **/dev/rmt/c***x***d***y***n**.

This option is useful when adding SCSI tape devices. For the **-r** option, both the **-d** and **-c** options are required.

For 68040-based machines, the **-t** option allows devices to be created of the type **/dev/tty***axx*, where *a* is an alphabetic character from a–z that signifies the expansion slot in which the TTY port is located, and *xx* is a two-digit decimal that signifies the TTY port number. For 68020-based machines, the **-t** option allows devices to be created of the type **/dev/tty***xxx*, where *xxx* is a three-digit decimal that signifies the TTY port number. For both types of machines, these are character-type devices, typically used for terminals and line printers, as well as other peripherals. For this option, the **-d** option is required.

The **-p** option allows devices to be created of the type **/dev/ttyp***xx*. These are character-type devices, typically used for virtual login sessions. An example of this is an Ethernet connection. The **-d** option is required.

FILES

/dev/tty*
/dev/ttyp*
/dev/dsk/*
/dev/rdsk/*
/dev/rmt/*

SEE ALSO

mknod(1M).

NAME

fsck, dfsck - check and repair file systems

SYNOPSIS

/etc/fsck [**-y**] [**-n**] [-sc:s] [**-s**] [-Sc:s] [**-S**] [**-t** file] [**-q**] [**-D**] [**-f**] [**-p**] [**-bB**] [**-O**] [**-M**] [**-E**] [file-systems]

/etc/dfsck [options1] filsys1 ... - [options2] filsys2 ...

DESCRIPTION

Fsck

fsck audits and interactively repairs inconsistent conditions for CTIX file systems. If the file system is consistent, the number of files, number of blocks used, and number of free blocks are reported. If the file system is inconsistent, the operator is prompted for concurrence before each correction is attempted. It should be noted that some corrective actions result in some loss of data: the amount and severity of data lost can be determined from the diagnostic output. The default action for each consistency correction is to wait for the operator to respond **yes** or **no**. If the operator does not have write permission, *fsck* defaults to an **-n** action. Upon completion, *fsck* reports the number of used and free 512-byte blocks and the number of files in the file system.

Modifying a mounted (**root**) file system requires special precautions by *fsck*, because a single *sync (2)* undoes all of *fsck's* repair work. To prevent this, *fsck* performs a *uadmin*(A_REMOUNT,0,0) [see *uadmin*(2)]. The system call forces CTIX to reread the super-block from the disk. If there is extensive damage to the mounted file system, *fsck* reboots CTIX.

The following options are interpreted by *fsck*:

-y Assumes a yes response to all *fsck* prompts.

-n Assumes a no response to all questions asked by *fsck* prompts; does not open the file system for writing.

-sc:*s*

-s Ignores the actual free list or bit map and (unconditionally) reconstructs a new one by rewriting the super-block of the file system. The file system should be unmounted while this is done; if this is not possible, care should be taken that the system is quiescent.

If *c*:*s* is given on a standard file system, the free list is organized with *c* blocks-per-cylinder and *s* blocks skipped. If *c*:*s* is omitted, the values originally specified to *mkfs* are used. If these values were not specified, the value *400*:*7* is used.

-S*c:s*

-S Conditionally reconstructs the free list or bit map. This option is like -s, described above, except that the free list or bit map is rebuilt only if no discrepancies are discovered in the file system. -S forces a no response to *fsck* prompts. This option is useful for forcing free list or bit map reorganization on uncontaminated file systems.

-t If *fsck* cannot obtain enough memory to keep its tables, it uses a scratch file. If the **-t** option is specified, the file named in the next argument is used as the scratch file, if needed. Without the **-t** flag, *fsck* prompts for the name of the scratch file. The file chosen should not be on the file system being checked, and if it is not a special file or did not already exist, it is removed when *fsck* completes.

-q Quiets *fsck*. Does not print size-check messages in Phase 1. Unreferenced FIFOs are silently removed. If *fsck* requires it, counts in the super-block are automatically fixed and the free list or bit map is salvaged.

-D Checks directories for consistency. This is useful after system crashes. The following inconsistencies are sought:

- Entries with null names but nonzero i-numbers.
- Entries that are not padded to full size with nulls.
- Invalid . and .. entries.
- Names that contain a slash (/).
- Final blocks that are not cleared past end-of-file.

-f Fast checks. Checks block and sizes (Phase 1) and checks the free list or bit map (Phase 5). The free list or bit map is reconstructed (Phase 6) if it is necessary.

-p Preens file systems only; intended for autoboot. The *fsck* program does not prompt for operator input; instead, it applies standard fixes whenever the fix doesn't involve loss of data. Only the following problems are subject to this kind of fix:

- Unreferenced i-nodes.
- Link counts in i-nodes too large.
- Missing blocks in the free list.
- Blocks in the free list also in files.
- Counts in the super-block wrong.

Any problem not of this type causes *fsck* to terminate with an error status. The startup script that runs *fsck* (normally **/etc/bcheckrc**) can specify the **-p** option to *fsck* and make a normal boot contingent upon a normal *fsck* return status.

-b or **-B**
If the file system being checked is the **root** file system and modifications have been made, this resyncs the file system, or reboots if necessary.

-E Converts file system to extended bit-map format.

-M Converts file system to new bit-map free list format.

-O Converts file system to old free list format.

The **-E**, **-M**, and **-O** options imply **-s**.

If no *file-systems* are specified, *fsck* reads a list of default file systems from the file **/etc/checklist**.

Inconsistencies are checked as follows:

1. Blocks claimed by more than one i-node or the free list.
2. Blocks claimed by an i-node or the free list outside the range of the file system.
3. Incorrect link counts.
4. Size checks:
   - Incorrect number of blocks.
   - Directory size not 16-byte aligned.
5. Bad i-node format.
6. Blocks not accounted for anywhere.
7. Directory checks:
   - File pointing to unallocated i-node.
   - I-node number out of range.
8. Super-block checks:
   - More than 65536 i-nodes.
   - More blocks for i-nodes than exist in the file system.
9. Bad free block list format.
10. Total free block and/or free i-node count incorrect.

Orphaned files and directories (allocated but unreferenced) are, with the operator's concurrence, reconnected by placing them in the **lost+found** directory, if the files are nonempty. The user is notified if the file or directory is empty or not. If it is empty, *fsck* silently removes them. The *fsck* program forces the reconnection of nonempty directories. The name assigned is the i-node number. The only restriction is that the directory **lost+found** must

preexist in the root of the file system being checked and must have empty slots in which entries can be made. The **lost+found** directory is normally created by running *mklost+found*(1M) just after the file system is created with *mkfs*(1M).

Checking the raw device is almost always faster and should be used with everything but the **root** file system.

Dfsck

The *dfsck* program allows two file system checks on two different drives simultaneously. *options1* and *options2* are used to pass options to *fsck* for the two sets of file systems. A dash (-) is the separator between the file system groups.

The *dfsck* program permits an operator to interact with two *fsck*(1M) programs at once. To aid in this, *dfsck* prints the file system name for each message to the operator. When answering a question from *dfsck*, the operator must prefix the response with a **1** or a **2** (indicating that the answer refers to the first or second file system group).

Do not use *dfsck* to check the **root** file system.

FILES

/etc/checklist	default list of file systems to check
/etc/checkall	optimizing *dfsck* shell file

SEE ALSO

clri(1M), init(1M), mklost+found(1M), uadmin(2), ncheck(1M), checklist(4), fs(4).
S/Series CTIX Administrator's Guide.

DIAGNOSTICS

The diagnostics produced by *fsck* are intended to be self-explanatory.

If **-p** was specified and preening was inadequate, a nonzero status is returned.

NOTES

Always unmount file systems before running *fsck* except in the case of the **root** file system.

The *block* device must be used with mounted file systems; thus, the **root** file system must always be specified as the block device.

The maintenance tape can be used to check the normal **root** file system as a raw device, unmounted. (In this case, the **root** file system is on the RAM disk.)

The *fsck* program determines the file system type (1K or 4K) on its own.

BUGS

I-node numbers for . and .. in each directory should be checked for validity.

The *fsck* program does not know how to create a **lost+found** directory.

NAME

fstyp - determine file system identifier

SYNOPSIS

fstyp *special*

DESCRIPTION

The *fstyp* command allows the user to determine the file system identifier of mounted or unmounted file systems using heuristic programs. The file system type is required by *mount*(2) and sometimes by *mount*(1M) to mount file systems of different types.

The directory **/etc/fstyp.d** contains a program for each file system type to be checked; each program applies some appropriate heuristic to determine whether the supplied *special* file is of the type for which it checks. If it is, the program prints on standard output the usual file-system identifier for that type and exits with a return code of 0; otherwise, it prints error messages on standard error and exits with a nonzero return code. The *fstyp* command runs the programs in **/etc/fstyp.d** in alphabetical order, passing *special* as an argument; if any program succeeds, its file-system type identifier is printed and *fstyp* exits immediately. If no program succeeds, *fstyp* prints "Unknown_fstyp" to indicate failure.

NOTE

fstyp reports "S51K" for both 1K and 4K System V file systems.

WARNING

The use of heuristics implies that the result of *fstyp* is not guaranteed to be accurate.

SEE ALSO

mount(1M), mount(2), sysfs(2).

NAME

hinv - hardware inventory

SYNOPSIS

/etc/hinv option

/etc/hinv hardware-item

DESCRIPTION

The *hinv* command provides hardware configuration information. There are two forms of the command. In the first form, an *option* is given and the result is printed on *stdout*; in the second form, a particular hardware item is specified, and *hinv* exits with **0** if it exists, or with **1** otherwise.

Option is one of the following:

-p Prints hardware configuration. Items are printed one per line.

-c Prints CPU type (68020 or 68040).

-f Prints FPU type (68040 or 68881). (For S/280 only: 68882 is reported as 68881.)

-s Prints system type.

-u Returns a meaningless value of 128; included for compatibility only.

-m Prints total physical memory in bytes.

Hardware-item is one of the following:

68881 68881 (on S/80, S/480, or S/640) or 68882 (on S/280) floating-point processor is present.

iop Terminal Accelerator Board is present.

422 Any RS-422 Cluster Board.

422-2 Two-channel RS-422 Cluster Board.

422-4 Four-channel RS-422 Cluster Board.

vme VME Interface Board.

s*n* RS-232 Board *n*.

scsi A SCSI interface is present.

S0 Onboard SCSI is present.

S*n* SCSI Combo Board *n*.

ipt	Interphase Tape Controller is in EEPROM.
smd	Interphase SMD Controller is in EEPROM.
mpcc	Multiprotocol Communications Controller is in EEPROM.
serial	Gives number of serial ports present.
disks	Gives number of disks present.
eeprom	**VME EEPROM** valid for **UNIX**.
enet	Ethernet Combo Board is present or a CMC Ethernet Board is in EEPROM.
cmcenp	CMC Ethernet Board is in EEPROM.
E*n*	Ethernet Combo Board *n*.
I*n*	IOP16 Board *n* is present (n = 1 .. 4).
hwfloat	Hardware floating point is present.

BUGS

The *hinv* command does not know about particular VME cards, which can be plugged in when the VME Interface Board is present.

NAME

includes - determine C language preprocessor include files

SYNOPSIS

includes [option ...] file ...

DESCRIPTION

The *includes* command determines the **#include** files necessary to compile a C language source file using the C language preprocessor *cpp*(1); the command is based on *cpp*(1) and takes the same options. (If you are running a 6.3 or later release of CTIX, *cpp* is a built-in feature of the C compiler.) Multiple source files can be named on the command line. However, instead of producing preprocessed code, *includes* produces on standard output a list of the **#include** file dependencies (directly or nested) of the named source files.

The output format is suitable for direct use in a *makefile* to be used by the *make*(1) command. For each named source file, the **#include** files are listed, one per line, preceded by the name of the source file (with the last letter of its name changed to the letter **o**). The two names are separated by a colon and a space : . For example, if source file **pgm.c** depends only on the **#include** file **incl.h**, the output of **includes** for the source file **pgm.c** would be

```
pgm.o: incl.h
```

The following *options* to *includes* are recognized:

-U*name* Removes any initial definition of *name*, where *name* is a reserved symbol that is predefined by the particular preprocessor. The current list of possibly-reserved symbols includes the following:

operating system:	ibm, gcos, os, tss, unix
hardware:	interdata, mc68k, mc68k32, m68k, mc68000, mc68010, mc68020 mc68881, mc68040, pdp11, u370, u3b, vax
system variants:	RES, RT

-D*name*

-D*name*=*def* Defines *name* with value *def* as if by a **#define**. If no =*def* is given, *name* is defined with value 1. The **-D** option has lower precedence than the **-U** option. That is, if the same name is used in both a **-U** option and a **-D** option, the name is undefined, regardless of the order of the options.

-T The **-T** option forces *includes* to use only the first eight characters to distinguish preprocessor symbols and is included for backward compatibility.

-I*dir* Changes the algorithm for searching for **#include** files whose names do not begin with / to look in *dir* before looking in the directories on the standard list. Thus, **#include** files with names enclosed in double quotation marks (" ") are searched for first in the directory of the file with the **#include** line, then in directories named in **-I** options, and last in directories on a standard list. For **#include** files with names enclosed in angle brackets (< >), the directory of the file with the **#include** line is not searched. By default, *includes* searches for the name enclosed in angle brackets in **/usr/include**; however, if the shell variable INCROOT is set, *includes* prepends the value of INCROOT to the standard list; this is particularly useful for cross-compilation.

-Y*dir* Uses directory *dir* in place of the standard list of directories when searching for **#include** files. The **-Y** option overrides the value for INCROOT if it is set.

-H Prints, one per line on standard error, the pathnames of included files.

Two special names are understood by *includes*: **__LINE__** is defined as the current line number (as a decimal integer) as known by *includes*, and **__FILE__** is defined as the current filename (as a C string) as known by *includes*. The special names can be used anywhere (including in macros), just as any other defined name.

All *cpp* directives understood by *includes* start with lines begun by #. The directives are as follows:

#define *name token-string*
Replaces subsequent instances of *name* with *token-string*.

#define *name*(*arg*, ..., *arg*) *token-string*
Notice that there can be no space between the *name* and the (. Replace subsequent instances of *name* followed by a (, a list of comma-separated tokens, and a) by *token-string* where each occurrence of an *arg* in the *token-string* is replaced by the corresponding token in the comma-separated list. When a macro with arguments is expanded, the arguments are placed into the expanded *token-string* unchanged. After the entire *token-string*

has been expanded, *includes* restarts its scan for names to expand at the beginning of the newly created *token-string*.

#ident *"string"*

This directive has no effect.

#undef *name* Causes the definition of *name* (if any) to be forgotten from now on.

#include *"filename"*
#include <*filename*>

Includes at this point the contents of *filename* (which is then run through *includes*). When the <*filename*> notation is used, *filename* is searched for only in the standard places; see the descriptions for the **-I** and **-Y** options for more detail.

#line *integer-constant "filename"*

This directive has no effect.

#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* The lines following are processed if and only if *name* has been the subject of a previous **#define** without being the subject of an intervening **#undef**.

#ifndef *name* The lines following are not processed if and only if *name* has been the subject of a previous **#define** without being the subject of an intervening **#undef**.

#if *constant-expression*

Lines following are processed if and only if the *constant-expression* evaluates to nonzero. All binary nonassignment C operators, the ?: operator, the unary –, !, and ~ operators are all legal in *constant-expression*. The precedence of the operators is the same as defined by the C language. There is also a unary operator **defined**, which can be used in *constant-expression* in these two forms: **defined** (*name*) or **defined** *name*. This allows the utility of **#ifdef** and **#ifndef** in a **#if** directive. Only these operators, integer constants, and names that are known by *includes* should be used in *constant-expression*. In particular, the **sizeof** operator is not available.

#elif *constant-expression*

An arbitrary number of **#elif** directives is allowed between a **#if**, **#ifdef**, or **#ifndef** directive and a **#else** or **#endif** directive. The lines following the **#elif** directive will appear in the output if and

only if the preceding test directive evaluates to zero, all intervening **#elif** directives evaluate to zero, and the *constant-expression* evaluates to nonzero. If *constant-expression* evaluates to nonzero, all succeeding **#elif** and **#else** directives will be ignored. Any *constant-expression* allowed in a **#if** directive is allowed in a **#elif** directive.

#else The lines following will appear in the output if and only if the preceding test directive evaluates to zero, and all intervening **#elif** directives evaluate to zero.

The test directives and the possible **#else** directives can be nested.

FILES

INCDIR	standard directory list for **#include** files, usually /usr/include
LIBDIR	usually /lib

SEE ALSO

cc(1), cpp(1), m4(1).

DIAGNOSTICS

The error messages produced by *includes* are intended to be self-explanatory. The line number and filename where the error occurred are printed along with the diagnostics.

NAME

iopdump - upload a Front-end I/O Processor's RAM

SYOPNSIS

/usr/local/bin/iopdump [**-p**] [**-i** iop16number] address length

DESCRIPTION

iopdump uploads and displays **length** number of memory data bytes beginning at **address** from a front-end I/O processor.

The **address** argument is a hexadecimal value from 0 to ffff.

The **length** argument is a decimal value.

The default front-end processor is an IOP.

The **-i** option specifies that the type of front-end I/O processor is an IOP16. For the **-i** option, the number **iop16number** must be a decimal number in the range 0 to 3. There is no default number.

The **-p** option causes the retrieved data to be printed as an ASCII hexadecimal dump to the standard output. Without this option, the binary data is sent to the standard output.

BUGS

For the IOP16, *iopdump* obtains only the data from the first board.

NAME

iv - initialize and maintain volume

SYNOPSIS

iv -iuostdwlvq special [descriptionfile]

DESCRIPTION

The *iv* command initializes and maintains a disk volume. *Special* and *descriptionfile*, described below, specify the disk and a description file for the disk volume. The *iv* command performs one of five operations, specified by the following options:

-i Completely initializes a volume. This consists of five phases:

1. Initializes *iv's* internal Volume Home Block, based on *descriptionfile* and the disk type. If the disk can support bad block handling, this creates an internal Bad Block Table. Puts bad block data from *descriptionfile* and volume's existing Bad Block Table (if any) in internal Bad Block Table.
2. Formats medium.
3. Performs a surface check. If the disk can support bad block handling, this adds bad blocks to the Bad Block Table. If the disk cannot support bad block handling, the first bad spot causes the disk to be rejected.
4. Writes out the Volume Home Block. This has the effect of dividing the volume into slices (partitions).
5. Allocates and writes out the files that share the Reserved Area (slice 0) with the Volume Home Block. If the disk can support bad block handling, one of these files is the Bad Block Table. Other files are specified in *descriptionfile*.

-u Updates the Volume Home Block. This is the same as **-i**, except that the second and third phases (medium formatting and surface check) are skipped.

-o Outputs a Volume Home Block and partition 0 to any file; requires a *descriptionfile*. The following command produces a dump tape:

```
iv -o /dev/rmt0 /usr/lib/iv/desc.tdump
```

-s Performs a surface test. Any bad blocks discovered are added to the Bad Block Table.

-t Tells volume description. Displays volume home block in human-readable form. No description file is needed. The volume's contents are not affected.

-d Displays description file. A description file that describes the current state of the volume is written to the standard output. If the Reserved Area contains a loader, the **loader** keyword's value is written as **/usr/lib/iv/loader**. If the Reserved Area contains a download image area, the Download Area Description lists files whose names are of the following form:

/usr/lib/iv/ws*xxx*.*yyy*

where *xxx* is the numeric device identification; *yyy* is **422** if *xxx* is even and **232** if *xxx* is odd.

The **-f** option, equivalent to **-u**, is provided for compatibility with older versions of *iv*. It should not be used, as it may disappear in future releases.

In addition to the single operation option (**-i**, **-u**, **-s**, **-t**, or **-d**) you can specify any or all of the following options:

-v Verbose display output. If the display includes the Volume Home Block, it also includes the Bad Block Table.

-l A normal surface test consists of a single pass over the disk; **-l** specifies ten passes.

-w A normal surface test pass consists of a read pass; **-w** specifies a write pass before each read pass.

-q Prints the size of the disk (in megabytes).

File Parameters

Special is the character special file for slice zero on the drive. This name takes the form **/dev/rdsk/c***n***d***t***s0**, where *n* is the controller number and *t* is the drive number.

Descriptionfile is a text file that describes the volume. It is required by the **-i** and **-u** options. The description file consists of five parts:

- General Description
- Reserved Area Description
- Bad Blocks Description
- Partition Table Description
- Download Area Description

Each description is separated from the next by a line that contains only a single dollar sign ($). Specifics for each of the five descriptions are given under separate headings below.

General Description

Each line in the General Description begins with a keyword. Some keywords are followed by values; the value is separated from the keyword by spaces or tabs. For example:

```
ecc
cylinders       1024
```

Each keyword is used only once; valid keywords follow:

type Mandatory unless the volume is already initialized in the appropriate format. Value is disk type: **HD** for onboard ST506 hard disk; **RD** for RAM disk; V3200 for SMD controller; SCSI for SCSI disk; and FD for floppy disk.

name Mandatory unless the volume is already initialized in the appropriate format. Value is the volume name. Any characters except spaces or tabs are permitted in the volume name; the serial number of the disk is the recommended volume name. The actual name in the Volume Home Block is always exactly six characters; *iv* right truncates names that are too long and right pads with nulls names that are too short.

cylinders Mandatory unless the volume is already initialized in the appropriate format. Value is the number of cylinders on the disk. For SCSI, cylinders X heads X sectors should be just less than twice the number of 1K logical blocks on the disk.

heads Mandatory unless the volume is already initialized in the appropriate format. Value is the number of heads on the disk.

sectors Mandatory unless the volume is already initialized in the appropriate format. Value is the number of physical sectors per track. The standard value is 32.

lsectors The number of logical 512-byte sectors on a SCSI disk. If this value is not supplied, the total number of available logical sectors on the device is used.

formatextra The SMD drive is formatted with an extra sector on each track. (This sector is ignored by CTIX but is required for some disk drives, notably the Eagle-XP.)

steprate Mandatory for ST506 unless the volume is already initialized in the appropriate format. Value is a number that is passed to the disk controller. The normal steprate for ST506 drives is 14; 0 can be used for slower drives. See the disk manufacturer's documentation for further information.

exchangeable If this keyword is present, the disk can be removed from its drive.

hitech (ST506 drives only) If this keyword is present, the reduced write current line to the disk is used for head-select bit 3 to allow more than eight heads.

precomp (ST506 drives only) The value is *c*/16, where *c* is the cylinder at which precompensation should start. See the disk manufacturer's documentation for further information.

ecc The disk has been prepared to function in ECC mode.

gap1
gap2 Gap size for SMD drives. See the disk manufacturer's documentation for further information.

unformattedbytes
The number of unformatted bytes per sector. This value is required if the "cyl head offset" format is used for the bad block table entries.

Reserved Area Description

The Reserved Area Description describes the files that share slice zero with the Volume Home Block. Each line in the Reserved Area Description consists of a keyword followed by one or more parameters; one or more tabs or spaces separate keywords and parameters from each other. Here are the valid keywords and their meanings. (A logical block is 1024 bytes long.)

loader Describes the loader area. The first (mandatory) parameter is the full pathname of an a.out file to put in the loader area. The second (optional) parameter is the size of the loader area in logical blocks. If the second parameter is missing, the size of the a.out file is used. The standard value is 128.

badblocktable Describes the Bad Block Table. The first (mandatory) parameter is the size of the Bad Block Table in logical blocks. The second (optional) parameter is only used when an existing Bad Block Table contains errors; this parameter is *empty* to clear the Bad Block Table, missing otherwise.

dump Describes area to contain dump after crash. The only (mandatory) parameter specifies the size of the dump area in logical blocks.

downloadarea Describes area to contain system images for downloading. The only (mandatory) parameter specifies the size of the download area in logical blocks. (The files actually put in this area are described separately. See the Download Area Description heading below.)

All lines valid for the Reserved Area Description are optional. However, the Bad Block Table is mandatory on a volume that supports bad block handling (other than SCSI); the loader area is mandatory on a volume that is to hold an operating system.

Bad Block Description

The Bad Block Description explicitly specifies up to 889 bad blocks to be added to the Bad Block Table. The *iv* command merges specified bad block information with information already in the Bad Block Table (if there already is one) and bad block information discovered through the surface test.

Each bad block entry is a single line. There are three forms:

sector

where *sector* is a physical sector number;

cylinder head offset

where *cylinder* is a cylinder number, *head* is a head number, and *offset* is the byte offset of the bad spot;

cylinder head sector

where *cylinder* is a cylinder number, *head* is a head number, and *sector* is a physical sector number of the bad spot. The third form is selected by placing the keyword **sector** on the line preceding the first entry of this type in the bad block description. Entries using the third form must come last in the list of bad blocks. All three forms condemn a single sector; the second form condemns the sector that contains the specified byte.

The last sector on each track serves as a bad block alternate. *iv* chooses the alternates in a way that minimizes extra seeking for alternate blocks.

Partition Table Description

The Partition Table Description specifies where the slices (partitions) on the disk are to begin and end. Each line in the description specifies the starting logical block of a slice. Start blocks must be on track boundaries, except in the

case of SCSI drives, where start blocks need only be on a logical block boundary.

Except for overlapping partitions, slices must be listed in ascending numeric order; the beginning of a slice defines the end of the previous slice.

It is possible to specify overlapping partitions, although care must be taken in doing so. A $ following any block number indicates that the slice extends to the end of the disk, beyond the next boundary number. Any slice with a starting block number that is larger than its successor must extend to the end of the disk (and must therefore be followed by the $ parameter).

For example, the following description specifies five slices; the fifth slice extends from the second slice to the end of the disk:

```
0
16
20016
40016   $
16      $
```

The first logical block boundary number in the description must always be 0. The last slice in the description always extends to the end of the disk ($ is optional).

There can be at most 16 slices on a disk.

It is a fatal error to specify a slice 1 that does not leave enough room in slice 0 for the Volume Home Block and the slice 0 files.

Download Area Description

The Download Area Description specifies system images to be included in the Download Area. Each line in the description consists of a numeric device identification and the full pathname of the file to be copied into the download area; the two parts of the line are separated by one or more spaces or tabs.

EXAMPLES

The following example shows a disk description file for a nonbootable disk (bad blocks expressed in "cylinder/head/sector" format):

```
#       MAXTOR 85 MB disk
type            HD
name            Serno
cylinders       1024
heads           8
sectors         17
steprate        14
hitech
```

```
ecc
$
badblocktable     1
$
sector
15          5          4
$
0
8
$
$
```

The following file describes a bootable SMD (bad blocks expressed in ''cylinder/head/offset'' format):

```
type                V3200
name                Serno
cylinders           1489
heads               11
sectors             33
ecc
gap1                16
gap2                16
unformattedbytes    620
$
badblocktable       3
dump                1024
downloadarea        300
loader              /usr/lib/iv/loader    128
$
12          3          405
187         9          1010
692         4          5228
66          2          657
985         5          3398
$
0
1456
17360
25360
```

```
45360
85360
125360
165360  $
$
100     /usr/lib/iv/ws100.422
200     /usr/lib/iv/ws200.422
$
```

The following file describes a bootable Hitachi drive (bad blocks expressed as physical sector numbers):

```
type            HD
name            Serno
cylinders       823
heads           10
sectors         17
steprate        14
hitech
ecc
$
badblocktable   1
dump            1024
downloadarea    300
loader          /usr/lib/iv/loader   128
$
1048
2441
5064
15119
15678
23533
23534
42091
43918
60466
60467
$
0
1456
```

```
17730
25922
46402   $
$
100     /usr/lib/iv/ws100.422
200     /usr/lib/iv/ws200.422
$
```

The following file describes a drive without a dump area (no bad blocks specified):

```
type            HD
name            Serno
cylinders       645
heads           7
sectors         17
steprate        14
precomp         80
hitech
ecc
$
badblocktable   1
downloadarea    300
loader          /usr/lib/iv/loader   128
$
$
0
432
12328
18328   $
$
100     /usr/lib/iv/ws100.422
200     /usr/lib/iv/ws200.422
$
```

FILES

/dev/rdsk/*	disk character special files
/usr/lib/iv/desc.*	prototype description files

SEE ALSO

disk(7).

S/Series CTIX Administrator's Guide.

"WD2010-05 Winchester Disk Controller" in *Storage Management Products Handbook.* Irvine, Calif.: Western Digital Corp., 1984.

NOTES

Any device in physical mode (for example, while surface testing or formatting is being done) is an exclusive open device: use the maintenance tape to reformat or run surface tests on the boot device.

A typical disk has fewer bad spots than the total number of megabytes (a 40-MB drive should have fewer than 40 bad spots, and so forth).

WARNINGS

The **-i**, **-u**, and **-s** operations are dangerous or fatal to existing volume data. Always precede these operations with a backup.

When a new bad block is itself an alternate block, *iv* may produce messages that appear spurious but are actually correct. If the bad block is already in use as an alternate, the message can appear twice for one block.

Do not run *mkfs*(1M) on an overlapping partition.

Do not use Partition Table Descriptions from pre-5.0 versions of CTIX that specify partitions by track numbers, rather than by logical block boundaries.

The system does not allow dumps of size greater than or equal to 64 MB.

NAME

machid: mc68k, miti, mini, mega, unixpc, i386, i286, pdp11, u3b, u3b2, u3b5, u3b15, u370, vax - get processor type truth value

SYNOPSIS

mc68k

miti

mini

mega

unixpc

i386

i286

pdp11

u3b

u3b2

u3b5

u3b15

u370

vax

DESCRIPTION

The following commands return a true value (exit code of 0) if you are on a processor that the command name indicates.

mc68k True if you are on a 68000-, 68010-, 68020-, or 68040-based computer.

miti True if you are on an S/Series computer.

mini True if you are on a MiniFrame computer.

mega True if you are on an S/1280 computer.

unixpc True if you are on a Unix PC computer.

i386 True if you are on an Intel 80386-based computer.

i286 True if you are on an Intel 80286-based computer.

pdp11 True if you are on a PDP-11/45 or PDP-11/70.

u3b True if you are on a 3B20 computer.

u3b2 True if you are on a 3B2 computer.

u3b5 True if you are on a 3B5 computer.

u3b15 True if you are on a 3B15 computer.

u370 True if you are on an IBM 370 computer.

vax True if you are on a VAX-11/750 or VAX-11/780.

The commands that do not apply will return a false (nonzero) value. These commands are often used within makefiles [see *make*(1)] and shell procedures [see *sh*(1)] to increase portability.

SEE ALSO

make(1), sh(1), test(1), true(1).

NAME

mcs - manipulate the object file comment section

SYNOPSIS

mcs [options] object-file ...

DESCRIPTION

The *mcs* command manipulates the comment section, normally the ''.comment'' section, in an object file. It is used to add, delete, print, and compress the contents of the comment section in a CTIX system object file. *mcs* must be given one or more of the options described below. It takes each of the options given and applies them in order to the *object-files*.

If the object file is an archive, the file is treated as a set of individual object files. For example, if the **-a** option is specified, the string is appended to the comment section of each archive element.

The following options are available.

-a *string*
: Appends *string* to the comment section of the *object-files*. If *string* contains embedded blanks, it must be enclosed in quotation marks.

-c
: Compresses the contents of the comment section. All duplicate entries are removed. The ordering of the remaining entries is not disturbed.

-d
: Deletes the contents of the comment section from the object file. The object file comment section header is removed also.

-n *name*
: Specifies the name of the section to access. By default, *mcs* deals with the section named *.comment*. This option can be used to specify another section.

-p
: Prints the contents of the comment section on the standard output. If more than one name is specified, each entry printed is tagged by the name of the file from which it was extracted, using the format ''filename:string.''

EXAMPLES

To print a file's *comment* section:

```
mcs -p file
```

To append *string* to *file*'s *comment* section:

```
mcs -a string file
```

FILES

/usr/tmp/mcs*	temporary files
/usr/tmp/*	temporary files

SEE ALSO

cpp(1), a.out(4).

NOTES

mcs cannot insert or delete comment sections in executable objects with magic number 0413. (By default, *ld* creates executable objects with magic number 0413 [see *a.out*(4) and *ld*(1)].) However, the **-d** option to *mcs* can make the comment section of a 0413 file zero-length. This allows use of *mcs* with the **-a** and **-c** options on such files. All libraries provided with CTIX have comment sections in each library member as well as in the C start-up routines; thus, there would normally be a comment section in every **a.out** file.

NAME

mkfs - construct a file system

SYNOPSIS

/etc/mkfs special [**-O**] [**-M**] [**-E**] blocks[:i-nodes] [gap blocks/cyl] [**-b** blocksize]

/etc/mkfs special [**-O**] [**-M**] [**-E**] proto [gap blocks/cyl] **-b** blocksize]

/etc/mkfs special [**-O**] [**-M**] [**-E**]

DESCRIPTION

The *mkfs* command constructs a file system by writing on the *special* file using the values found in the remaining arguments of the command line (except in the case of the third form of the command, discussed below). The command waits 10 seconds before starting to construct the file system. During this 10-second pause, the command can be aborted with a delete character.

The **-b** *blocksize* option specifies the logical block size for the file system: the number of bytes read or written by the operating system in a single I/O operation. Valid values for *blocksize* are 1024 and 4096. If the **-b** option is omitted, the default block size is 1024.

Note that if you make a 4K file system, you must add the **buffers_4k** parameter in the **dfile** to specify the number of 4K file system buffers to use.

If the second argument is a string of digits, the size of the file system is the value of *blocks* interpreted as a decimal number. This is the number of *physical* (512-byte) disk blocks the file system occupies. If the number of i-nodes is not given, the default is the number of *logical* (1024- or 4096-byte) blocks divided by 4 (rounded down); i-nodes are allocated in groups of 16. The *mkfs* command builds a file system with a single empty directory on it. The boot program block (block zero) is left uninitialized.

If the second argument is the name of a file that can be opened, *mkfs* assumes it to be a prototype file *proto*, and takes its directions from that file. The prototype file contains tokens separated by spaces or new-lines. A sample prototype specification follows (line numbers are added for clarity):

```
1       /stand/diskboot
2       4872 110
3       d--777 3 1
4       usr     d--777 3 1
5               sh      ---755 3 1 /bin/sh
6               ken     d--755 6 1
7                       $
```

```
8               b0      b--644 3 1 0 0
9               c0      c--644 3 1 0 0
10              $
11      $
```

Line 1 in the example is the name of a file to be copied onto block zero as the bootstrap program.

Line 2 specifies the number of *physical* (512-byte) blocks the file system is to occupy and the number of i-nodes in the file system.

Lines 3 through 9 tell *mkfs* about files and directories to be included in this file system.

Line 3 specifies the **root** directory.

Lines 4 through 6, 8, and 9 specify other directories and files.

The dollar sign ($) on line 7 instructs *mkfs* to end the branch of the file system it is on and continue from the next higher directory. The dollar signs on lines 10 and 11 end the process, since no additional specifications follow.

File specifications give the mode, the user ID, the group ID, and the initial contents of the file. Valid syntax for the contents field depends on the first character of the mode.

The mode for a file is specified by a six-character string. The first character specifies the type of the file. The character range is **-bcd** to specify regular, block special, character special, and directory files, respectively. The second character of the mode is either **u** or -, to specify set-user-ID mode or not. The third character is **g** or -, for the set-group-ID mode. The rest of the mode is a three-digit octal number specifying the owner, group, and other read, write, and execute permissions [see *chmod*(1)].

Two decimal number tokens come after the mode; they specify the user and group IDs of the owner of the file.

If the file is a regular file, the next token of the specification can be a pathname to where the contents and size are copied. If the file is a block or character special file, two decimal numbers follow, which give the major and minor device numbers. If the file is a directory, *mkfs* makes the entries . and .. and then reads a list of names and (recursively) file specifications for the entries in the directory. As noted above, the scan is terminated with the token dollar sign ($).

The first two forms of the command allow the rotational *gap* and the number of *blocks/cyl* to be specified. The default gap size is 7. The default blocks/cylinder is 400. The default is used if the supplied *gap* and *blocks/cyl* are considered illegal values, or if a short argument count occurs.

The **-O** option makes a file system with a free list instead of a bit map. This is the default for removeable disks.

The **-M** option makes a file system with a bit map in addition to a free list. This is the default for fixed disks.

The **-E** option makes a file system with a bit map only one file system.

Special must be a disk slice. The third form of the *mkfs* command extracts the slice size from the Volume Home Block and creates a file system the same size; this third option cannot be used where there are overlapping partitions. The number of i-nodes is the number of logical blocks divided by 4. Optimal values for gap size and blocks/cylinder are calculated; these may not be 7 and 400.

SEE ALSO

chmod(1), dir(4), fs(4).
S/Series CTIX Administrator's Guide.

BUGS

With a prototype file, there is no way to specify links. The maximum number of i-nodes configurable is 65500.

NAME

mount, umount - mount and unmount file systems and remote resources

SYNOPSIS

/etc/mount

/etc/mount directory

/etc/mount [**-r**] [**-f** fstyp] special directory

/etc/mount [**-r**] **-f NFS** [,options] special directory

/etc/mount [**-r**] [**-c**] **-d** resource directory

/etc/umount special

/etc/umount directory

/etc/umount [**-d**] resource

DESCRIPTION

File systems other than **root** (/) are considered *removable* in the sense that they can be either available to users or unavailable. The *mount* command announces to the system that *special* (a block special device) or *resource* (a remote resource) is available to users from the mount point *directory*. Note that *directory* must already exist; it becomes the name of the root of the newly mounted *special* or *resource*. A unique resource can be mounted only once (no multiple mounts).

When invoked with no arguments, *mount* displays the entire mount table. When entered with arguments, *mount* adds an entry to the table of mounted devices, **/etc/mnttab**. The *umount* command removes the entry. If invoked with any of the following partial argument lists, *mount* searches **/etc/fstab** for the missing arguments: *special*, **-d** *resource*, *directory*, or **-d** *directory*. No special syntax is required to mount a 4K file system.

The following options are available:

-v With no other arguments, prints a more verbose mount table containing file system type identifier (S51K, DUFST, NFS); with other arguments, prints the fully expanded *mount* command before mounting.

-r Indicates that *special* or *resource* is to be mounted read-only. If *special* or *resource* is write-protected or read-only advertised, this flag must be used.

-c Indicates that remote reads and writes should not be cached in the local buffer pool. **-c** is used in conjunction with **-d**.

-d Indicates that *resource* is a remote resource to be mounted on *directory* or unmounted. To mount a remote resource, Remote File Sharing (RFS) or the Network File System (NFS) must be up and running, and the resource must be advertised by a remote computer [see *rfstart*(1M) and *adv*(1M)]. If **-d** is not used, and this is not an NFS mount, *special* must be a local block special device.

-f *fstyp* Indicates that *fstyp* is the file system type to be mounted. If this argument is omitted, it defaults to the **root** *fstyp*.

If *fstyp* is NFS, NFS comma-separated options can be added after the *fstyp*. The available NFS options follow:

soft	Returns error if the server does not respond.
rsize=*n*	Sets the read-buffer size to *n* bytes.
wsize=*n*	Sets the write-buffer size to *n* bytes.
timeo=*n*	Sets the initial NFS timeout to *n* tenths of a second.
retrans=*n*	Sets the number of NFS retransmissions to *n*.
port=*n*	Sets the server IP port number to *n*.

special Indicates the block special device to be mounted on *directory*. If *fstyp* is NFS, *special* should be of the form *hostname*:*/pathname*.

resource Indicates the remote resource name to be mounted on *directory*.

directory Indicates the directory mount point for *special* or *resource*. (The directory must already exist.)

The *umount* command announces to the system that the file system previously mounted *special* or *resource* is to be made unavailable. If invoked with an incomplete argument list, *umount* searches **/etc/fstab** for the missing arguments.

Note that *mount* can be used by any user to list mounted file systems and resources. Only the superuser can mount and unmount file systems.

FILES

/etc/mnttab mount table
/etc/fstab file system table

SEE ALSO

adv(1M), fuser(1M), mountd(1M), nfsd(1M), nsquery(1M), rfstart(1M), rmntstat(1M), setmnt(1M), showmount(1M), unadv(1M), mount(2), umount(2), fstab(4), mnttab(4).
S/Series CTIX Administrator's Guide.

DIAGNOSTICS

If the *mount*(2) system call fails, *mount* prints an appropriate diagnostic. The *mount* command issues a warning if the file system to be mounted is currently mounted under another name. A remote resource mount fails if the resource is not available, or if it is advertised read-only and not mounted with **-r**, or if Remote File Sharing is not running.

The *umount* command fails if *special* or *resource* is not mounted or if it is busy. *special* or *resource* is busy if it contains an open file or some user's working directory. In such a case, use *fuser*(1M) to list and kill processes using *special* or *resource*.

NAME

occ - old C compiler

SYNOPSIS

occ [options] files

DESCRIPTION

The *occ* command is the interface to the pre-6.3 CTIX C Compilation System. The compilation tools consist of a preprocessor, compiler, optimizer, assembler, and link editor. The *occ* command processes the supplied options and then executes the various tools with the proper arguments. The *occ* command accepts several types of files as arguments.

Files whose names end with **.c** are taken to be C source programs and may be preprocessed, compiled, optimized, assembled, and link edited. The compilation process may be stopped after the completion of any pass if the appropriate options are supplied. If the compilation process runs through the assembler, then an object program is produced and is left in the file whose name is that of the source with **.o** substituted for **.c**. However, the **.o** file is normally deleted if a single C program is compiled and then immediately link edited. In the same way, files whose names end in **.s** are taken to be assembly source programs, and may be assembled and link edited; files whose names end in **.i** are taken to be preprocessed C source programs and may be compiled, optimized, assembled, and link edited. Files whose names do not end in **.c, .s,** or **.i** are handed to the link editor.

Since the *occ* command usually creates files in the current directory during the compilation process, it is necessary to run the *occ* command in a directory in which a file can be created. The following options are interpreted by *occ*.

-#
-##
-### These options cause *occ* to display each command that it would generate if it were to execute, but to fully execute only in the case of **-#**. Thus, **-#** specifies execution in verbose mode; **-##** specifies verbose mode (what *occ* would do if it were to execute) and checks permissions on all necessary files, but does not compile; and **-###** specifies verbose mode (what *occ* would do if it were to execute), but does nothing.

-c Suppresses the link editing phase of the compilation, and does not remove any produced object files.

-g Causes the compiler to generate additional information needed for the *sdb* (1).

-o *outfile*

Produces an output object file by the name *outfile*. The name of the default file is **a.out**. This is a link editor option.

-p Arranges for the compiler to produce code that counts the number of times each routine is called; also, if link editing takes place, profiled versions of libc.a and libm.a (with -lm option) are linked and *monitor*(3C) is automatically called. A **mon.out** file will then be produced at normal termination of execution of the object program. An execution profile can then be generated by *prof*(1).

-w Tells the linker (*ld*) not to print warnings about symbols that are partially matched. This option is meaningful only when the **-T** option is also specified.

-B*string*

-t[p02al]

These options will be removed in the next release. Use the **-Y** option.

-E Runs only *ocpp* on the named C programs, and sends the result to the standard output.

-H Prints out on *stderr* the pathname of each file included during the current compilation.

-O Compiles with optimization. This option will not have any effect on **.s** files.

-P Runs only *ocpp* on the named C programs and leaves the result in corresponding files suffixed **.i**. This option is passed to *ocpp*.

-S Compiles and does not assemble the named C programs; leaves the assembler-language output in corresponding files suffixed **.s**.

-T Truncates variable names to eight characters. Tells the loader to match eight character names (same as **-G** in the loader).

-Wc,arg1[,arg2...]

Hands off the argument[s] *argi* to pass *c* where *c* is one of [**p02al**] indicating the preprocessor, compiler, optimizer, assembler, or link editor, respectively. For example: **-Wa,-m** passes **-m** to the assembler.

-Y [p02alSILUc], *dirname* | *processor*

Specifies a new pathname, *dirname*, for the locations of the tools and directories designated in the first argument; or selects a processor type, *processor*, for which to generate code. **[p02alSILUc]** represents:

- **p** preprocessor
- **0** compiler
- **2** optimizer
- **a** assembler
- **l** link editor
- **S** directory containing the start-up routines
- **I** default include directory searched by *ocpp*
- **L** first default library directory searched by *old*
- **U** second default library directory searched by *old*
- **c** selects the processor type, specified by the second argument, for which to generate code: **68020**, **68010**, and **68881**. For example, **-Y c,68020** selects the 68020 processor with software floating-point instructions. Note that **68881** implies **68020**.

If the location of a tool is being specified, then the new pathname for the tool will be *dirname/tool*. If more than one **-Y** option is applied to any one tool or directory, then the last occurrence holds.

The *occ* command also recognizes **-C**, **-D**, **-H**, **-I**, and **-U** and passes these options and their arguments directly to the preprocessor without using the **-W** option. Similarly, the *occ* command recognizes **-l**, **-m**, **-o**, **-r**, **-s**, **-t**, **-u**, **-w**, **-x**, **-z**, **-F**, **-G**, **-L**, **-M**, **-N**, **-V**, and **-Z** and passes these options and their arguments directly to the loader. See the *cpp*(1) and *ld*(1) man pages for descriptions.

Other arguments are taken to be C-compatible object programs, typically produced by an earlier *occ* run, or perhaps libraries of C-compatible routines, and are passed directly to the link editor. These programs, together with the results of any compilations specified, are link edited (in the order given) to produce an executable program with name **a.out** unless the **-o** option of the link editor is used.

The *occ* command uses a filename *prefix*cc, so the prefix is parsed off the command and used to call the tools, that is, *prefix*tool. For example, OLDcc will call OLDcpp, OLDcomp, OLDoptim, OLDas, and OLDld and will link OLDcrt1.o. Therefore, you MUST be careful when moving the *occ* command around. The prefix will apply to the preprocessor, compiler, optimizer, assembler, link editor, and the start-up routines.

The C compiler uses one of three code generators for the 68010, 68020, and 68020/68881. There are several ways to select a particular code generator, but the selection is normally done using one of two basic mechanisms.

The first is to specify the processor on the *occ* command line, for example, by using the **-Y** option. (An equivalent mechanism is provided by the *gencc*(1) command, and also by the *cc1sw*(1), *cc2sw*, or *cc2fp* commands.) The **-Y** option has additional arguments that allow you to specify pathnames of default libraries, include files, and tools as described earlier.

The second mechanism is to use the CENVIRON shell variable. Note that the first mechanism, specifying the processor and/or search path of libraries and include files, overrides the CENVIRON and any other shell variable settings.

The CENVIRON variable has the following syntax:

CPU=xxxxx,FPU=yyyyy

where CPU indicates the central processor to generate for and FPU indicates the style of floating-point math to use. *xxxxx* may be 68010 or 68020, and *yyyyy* may be 68881 or SOFTWARE. The FPU parameter may be omitted; the default is SOFTWARE. The CENVIRON variable should always be set to the appropriate values in the **.profile** or **.cshrc** files or in the makefile. [See *hinv*(1M).]

The C compiler interprets two shell variables that, along with the CENVIRON variable, allow cross-compilation for any CTIX machine:

LIBROOT — This variable is a path that is prepended to normal library names when searching for a library. See also *ld*(1).

INCROOT — This variable is a path that is prepended to the **/usr/include** and **/usr/include/sys** directories during include file searches. See also *cpp*(1).

The C language standard was extended to allow arbitrary length variable names. The option pair "**-Wp,-T -W0,-XT**" will cause *occ* to truncate arbitrary length variable names to eight characters.

FILES

file.c	C source file
file.i	preprocessed C source file
file.o	object file
file.s	assembly language file
a.out	link edited output
LIBDIR/*ocrt1.o	start-up routine
LIBDIR/ocrtn.o	start-up routine

TMPDIR/*	temporary files
LIBDIR/ocpp	preprocessor, *cpp*(1)
LIBDIn/occom	68010 compiler
LIBDIR/occom20	68020 compiler
LIBDIR/occom20.81	68020/68881 compiler
LIBDIR/ooptim	optimizer
BINDIR/oas	assembler, *as*(1)
BINDIR/old	link editor, *ld*(1)
LIBDIR/libc.a	standard C library
LIBDIR/libc_s.a	standard C shared library

LIBDIR is usually /lib
BINDIR is usually /bin
TMPDIR is usually /tmp but can be redefined by setting the environment variable **TMPDIR** [see *tempnam*( ) in *tmpnam*(3S)].

SEE ALSO

as(1), ld(1), cc1sw(1), cpp(1), gencc(1), lint(1), prof(1), sdb(1), tmpnam(3S).
Programmer's Guide: CTIX Supplement.
Kernighan, B.W., and Ritchie, D.M., *The C Programming Language*, Prentice-Hall, 1978.

CAVEATS

occ will complain if it encounters inconsistencies between the processor selected and default libraries or include files.

Sometimes the range for a branch does not fit inside a word; in this case, an error message is printed. For suggested workarounds, see the section called "Span-Dependent Optimization" in Chapter 14 of the *Programmer's Guide: CTIX Supplement.*

DIAGNOSTICS

The diagnostics produced by the C compiler are sometimes cryptic. Occasional messages may be produced by the assembler or link editor.

NOTES

By default, the return value from a compiled C program is completely random. The only two guaranteed ways to return a specific value is to explicitly call *exit*(2) or to leave the function **main**( ) with a "*return expression;*" construct.

NAME

passmgmt - password files management

SYNOPSIS

passmgmt -a options name
passmgmt -m options name
passmgmt -d name

DESCRIPTION

The *passmgmt* command updates information in the password files. This command works with both **/etc/passwd** and **/etc/shadow**. If there is no shadow password file, the changes done by *passmgmt* will go in **/etc/passwd**.

The *passmgmt -a* form of the command adds an entry for user *name* to the login password files. This command does not create any directory for the new user and the new login remains locked (with the string ***LK*** in the *password* field) until the *passwd*(1) command is executed to set the password.

The *passmgmt -m* form of the command modifies the entry for user *name* in the login password files. The name field in the **/etc/shadow** entry and all the fields (except the password field) in the **/etc/password** entry can be modified by this command. Only fields entered on the command line are modified. If there is no **/etc/shadow** file, all modifications are made in **/etc/passwd**.

The *passmgmt -d* command deletes the entry for user *name* from the login password files. It does not remove any files the user owns on the system; they must be removed manually.

The login name of the user, *name*, must be unique.

The following options are available:

-c *comment* A short description of the login. It is limited to a maximum of 128 characters and defaults to an empty field. If the comment is more than one word, it must be enclosed in single or double quotation marks.

-h *homedir* Home directory of *name*. It is limited to a maximum of 256 characters and defaults to **/u/***name*.

-u *uid* UID of the *name*. This number must range from 0 to the maximum value for the system. It defaults to the next available UID greater than 100. Without the **-o** option, it enforces the uniqueness of a UID.

-o This option allows a UID to be nonunique. It is used only with the **-u** option.

-g *gid* GID of the *name*. This number must range from 0 to the maximum value for the system. The default is 1.

-s *shell* Login shell for *name*. It should be the full pathname of the program to be executed when the user logs in. The maximum length of *shell* is 256 characters. The default is for this field to be empty and to be interpreted as **/bin/sh**.

-l *logname* This option changes the *name* to *logname* for the **-m** option only.

The total size of each login entry, whether existing or new, is limited to a maximum of 511 bytes in the password files.

FILES

/etc/passwd
/etc/shadow
/etc/opasswd
/etc/oshadow

SEE ALSO

passwd(1), passwd(4).

DIAGNOSTICS

The *passmgmt* command exits with one of the following values:

0 Success.

1 Permission denied.

2 Invalid command syntax. Usage message of the *passmgmt* command is displayed.

3 Invalid argument provided to option.

4 UID in use.

5 Inconsistent password files (for example, *name* is in the **/etc/passwd** file and not in the **/etc/shadow** file, or vice versa).

6 Unexpected failure. Password files unchanged.

7 Unexpected failure. Password file(s) missing.

8 Password file(s) busy. Try again later.

9 *name* does not exist (if **-m** or **-d** is specified), already exists (if **-a** is specified), or *logname* already exists (if **-m -l** is specified).

NOTE

Do not use a colon or carriage return; these characters are interpreted as field separators.

NAME

passwd - change login password

SYNOPSIS

passwd [name]

passwd -s [name]

passwd -l [**-f**] [**-x** max] [**-n** min] name

passwd -d [**-f**] [**-x** max] [**-n** min] name

passwd -s [**-a**]

DESCRIPTION

The *passwd* command changes, sets, or lists attributes of a password associated with the login *name*. Ordinary users can change only the password that corresponds to their login *name*; the superuser can additionally set or change passwords and attributes associated with any login *name*.

When used to change a password, *passwd* prompts ordinary users for their old password, if any; it then prompts for the new password twice. When the old password is entered, *passwd* checks to see if the old password has "aged" sufficiently. Password "aging" is the amount of time (usually a number of days) that must elapse between password changes. If aging is insufficient, *passwd* terminates; see *passwd*(4).

Assuming aging is sufficient, a check is made to ensure that the new password meets construction requirements. When the new password is entered a second time, the two copies of the new password are compared. If the two copies are not identical, the cycle of prompting for the new password is repeated (at most, two more times).

Passwords must be constructed to meet the following requirements:

- Each password must have at least six characters. Only the first eight characters are significant.
- Each password must contain at least two alphabetic (upper- and lowercase) characters and at least one numeric or special character.
- Each password must differ from the user's login *name* and any reverse or circular shift of that login *name*. For comparison purposes, an uppercase letter and its corresponding lowercase letter are equivalent.
- New passwords must differ from the old by at least three characters. For comparison purposes, an uppercase letter and its corresponding lowercase letter are equivalent.

One whose effective user ID is zero is called a superuser; see *id*(1) and *su*(1). Superusers can change any password; therefore, *passwd* does not prompt superusers for the old password. Superusers are not forced to comply with password aging and password construction requirements. A superuser can create a null password by entering a carriage return in response to the prompt for a new password.

Any user can use the **-s** option to show password attributes for the login *name*.

The format of the display is as follows:

name status mm/dd/yy min max

or, if password aging information is not present,

name status

where:

name Is the login ID of the user.

status Is the password status of *name*: PS stands for passworded or locked, LK stands for locked, and NP stands for no password.

mm/dd/yy Is the date the password was last changed for *name*. Note that all password aging dates are determined by using Greenwich Mean Time and, therefore, may differ by as much as a day in other time zones.

min Is the minimum number of days required between password changes for *name*.

max Is the maximum number of days the password is valid for *name*.

Only a superuser can use the following options:

-l Locks the password entry for *name*.

-d Deletes the password for *name*. The login *name* is not prompted for a password.

-n Sets the minimum field for *name*. The *min* field contains the minimum number of days between password changes for *name*. If *min* is greater than *max*, the user cannot change the password. Always use this option with the **-x** option, unless *max* is set to -1 (aging disabled), in which case, *min* need not be set.

-x Sets the maximum field for *name*. The *max* field contains the number of days the password is valid for *name*. The aging for

name is disabled immediately if *max* is set to -1. If *max* is set to 0, the user must change the password at the next login session, and aging is disabled.

-a Shows password attributes for all entries. Use only with the **-s** option; *name* must not be provided.

-f Forces the user to change the password at the next login by expiring the password for *name*.

FILES

/etc/passwd
/etc/shadow
/etc/opasswd
/etc/oshadow

SEE ALSO

id(1M), login(1), passmgmt(1M), pwconv(1M), su(1M), crypt(3C), passwd(4).

DIAGNOSTICS

The *passwd* command exits with one of the following values:

0 Success.

1 Permission denied.

2 Invalid combination of options (incorrect syntax).

3 Unexpected failure. Password file unchanged.

4 Unexpected failure. Password file(s) missing.

5 Password file(s) busy. Try again later.

6 Invalid argument to option.

WARNINGS

If the optional **/etc/shadow** file exists, *passwd* uses that file instead of **/etc/passwd** to obtain password information. Because the two files store password aging information in different ways, the output from the *passwd* options can differ.

NAME

profiler: prfld, prfstat, prfdc, prfsnap, prfpr - operating system profiler

SYNOPSIS

/etc/prfld [namelist]

/etc/prfstat on

/etc/prfstat off

/etc/prfdc file [period [off_hour]]

/etc/prfsnap file

/etc/prfpr file [cutoff [namelist]]

S/4040, S/640, S/480, S/280, and S/80 only:
/etc/prfstat time

DESCRIPTION

The *prfld*, *prfstat*, *prfdc*, *prfsnap*, and *prfpr* programs form a system of programs to facilitate an activity study of the CTIX operating system. A kernel configured with kernel profiling must be used: the **pfr** driver may be loaded with *lddrv*(1M).

The *prfld* program is used to initialize the recording mechanism in the system. It generates a table containing the starting address of each system subroutine as extracted from *namelist*.

The *prfstat* program is used to enable or disable the sampling mechanism. Profiler overhead is less than one percent as calculated for 500 text addresses. Note that *prfstat* also reveals the number of text addresses being measured.

Addresses are sampled every clock tick (definition for *Hz* is given in *param.h*). S/4040, S/640, S/480, S/280, and S/80 systems allow sampling every *time* microsecond: the lower limit is 100 microsecond intervals.

The *prfdc* and *prfsnap* programs perform the data collection function of the profiler by copying the current value of all the text address counters to a file where the data can be analyzed. The *prfdc* program stores the counters into *file* every *period* minutes and turns off at *off_hour* (valid values for *off_hour* are **0-24**). The *prfsnap* program collects data at the time of invocation only, appending the counter values to *file*.

The *prfpr* program formats the data collected by *prfdc* or *prfsnap*. Each text address is converted to the nearest text symbol (as found in *namelist*) and is printed if the percent activity for that range is greater than *cutoff*.

FILES

/dev/prf	interface to profile data and text addresses
/etc/lddrv/unix.exec	default for namelist file

SEE ALSO

prf(7).

NAME

pwconv - install and update /etc/shadow with information from /etc/passwd

SYNOPSIS

pwconv

DESCRIPTION

The *pwconv* command creates and updates **/etc/shadow** with information from **/etc/passwd**. If the **/etc/shadow** file does not exist, *pwconv* creates it with information from **/etc/passwd**. The command populates **/etc/shadow** with the user's login name, password, and password aging information. If password aging information does not exist in **/etc/passwd** for a given user, none is added to **/etc/shadow**; however, the "last changed" information is always updated.

If the **/etc/shadow** file does exist, the following tasks are performed:

- Entries in the **/etc/passwd** file but not in the **/etc/shadow** file are added to the **/etc/shadow** file.
- Entries in the **/etc/shadow** file but not in the **/etc/passwd** file are removed from **/etc/shadow**.
- Password attributes (such as password and aging information) that exist in an **/etc/passwd** entry are moved to the corresponding entry in **/etc/shadow**.

The *pwconv* program is a privileged system command that can be executed only by the superuser. The *passwd* command should be used to add or change password aging information or passwords.

FILES

/etc/passwd
/etc/shadow
/etc/opasswd
/etc/oshadow

WARNING

If malformed lines are encountered in **/etc/passwd** or **/etc/shadow**, these files may be truncated to the last good entry.

SEE ALSO

passwd(1M), passmgmt(1M), pwunconv(1M).

DIAGNOSTICS

The *pwconv* command exits with one of the following values:

0	Success.
1	Permission denied.
2	Invalid command syntax.
3	Unexpected failure. Conversion not done.
4	Unexpected failure. Password file(s) missing.
5	Password file(s) busy. Try again later.

NAME

sar - system activity reporter

SYNOPSIS

sar [**-ubdycwaqvmprDSAC**] [**-o** file] t [n]

sar [**-ubdycwaqvmprDSAC**] [**-s** time] [**-e** time] [**-i** sec] [**-f** file]

DESCRIPTION

In the first instance, *sar* samples cumulative activity counters in the operating system at *n* intervals of *t* seconds, where *t* should be 5 or greater. If the **-o** option is specified, it saves the samples in *file* in binary format. The default value of *n* is 1. In the second instance, with no sampling interval specified, **sar** extracts data from a previously recorded *file*, either the one specified by the **-f** option, or by default, the standard system activity daily data file **/usr/adm/sa/sa***dd* for the current day *dd*. The starting and ending times of the report can be bounded by use of the **-s** and **-e** *time* arguments of the form *hh*[:*mm*[:*ss*]]. The **-i** option selects records at *sec* second intervals. Otherwise, all intervals found in the data file are reported.

In either case, subsets of data to be printed are specified by the following options:

-u Reports CPU utilization (the default):

%usr, %sys, %wio, %idle
: Portion of time running in user mode, running in system mode, idle with some process waiting for block I/O, and otherwise idle. When used with **-D**, **%sys** is split into percentage of time servicing requests from remote machines (**%sys remote**) and all other system time (**%sys local**).

-b Reports buffer activity:

bread/s, bwrit/s Transfers per second of data between system buffers and disk or other block devices.

lread/s, lwrit/s Accesses of system buffers.

%rcache, %wcache
: Cache-hit ratios: that is, (1-bread/lread) as a percentage.

pread/s, pwrit/s Transfers through raw (physical) device mechanism. When used with **-D**, buffer caching is reported for locally-mounted remote resources.

-d Reports activity for each block device; for example, disk or tape drive. When data is displayed, the device specification *dsk-* is generally used to represent a disk drive. The following activity data is reported:

%busy, avque Portion of time device was busy servicing a transfer request, and average number of requests outstanding during that time.

r+w/s, blks/s Number of data transfers from or to device, and number of bytes transferred in 512-byte units.

avwait, avserv Average time in milliseconds that transfer requests wait idly on queue, and average time to be serviced (which for disks includes seek, rotational latency, and data transfer times). RS-422 activity is also reported in this section.

-y Reports TTY device activity:

rawch/s, canch/s, outch/s
Input character rate, input character rate processed by canon, and output character rate.

rcvin/s, xmtin/s, mdmin/s
Receive, transmit, and modem interrupt rates.

-c Reports system calls:

scall/s System calls of all types.

sread/s, swrit/s, fork/s, exec/s
Specific system calls.

rchar/s, wchar/s Characters transferred by read and write system calls. When used with **-D**, the system calls are split into strictly local calls, remote outgoing (client) calls, and remote incoming (server) calls.

-w Reports system swapping and switching activity:

swpin/s, swpot/s, bswin/s, bswot/s
Number of transfers and number of 512-byte units transferred for swapins and swapouts (including initial loading of some programs).

pswch/s Process switches.

-a Reports use of file access system routines: **iget/s, namei/s,**. and **dirblk/s**.

-q Reports average queue length while occupied, and percentage of time occupied:

runq-sz, %runocc
Run queue of processes in memory and runnable.

swpq-sz, %swpocc
Swap queue of processes swapped out but ready to run.

-v Reports status of process, i-node, and file tables:

proc-sz, inod-sz, file-sz, lock-sz, fhdr-sz
Entries/size for each table, evaluated once at sampling point.

ov Overflows that occur between sampling points for each table.

-m Reports message and semaphore activities:

msg/s, sema/s Primitives per second.

-p Reports paging activities:

vflt/s Address translation page faults (valid page not in memory).

pflt/s Page faults from protection errors (illegal access to page) or copy-on-writes.

pgfil/s vflt/s satisfied by page-in from file system.

rclm/s Valid pages reclaimed for free list.

-r Reports unused memory pages and disk blocks:

freemem Average pages available to user processes.

freeswap Disk blocks available for process swapping.

-D Reports Remote File Sharing (RFS) activity. When used in combination with **-u**, **-b** or **-c**, it causes **sar** to produce the remote file sharing version of the corresponding report. **-Du** is assumed when only **-D** is specified.

-S Reports server and request queue status:

serv/lo-hi Average number of Remote File Sharing servers on the system.

request %busy Percentage of time receive descriptors are on the request queue.

request avg lgth Average number of receive descriptors waiting for service when queue is occupied.

server %avail Percentage of time there are idle servers.

server avg avail Average number of idle servers when idle ones exist.

-A Reports all data. Equivalent to **-udqbwcayvmprSDC**.

-C Reports Remote File Sharing (RFS) buffer caching overhead:

snd-inv/s Number of invalidation messages per second sent by your machine as a server.

snd-msg/s Total outgoing RFS messages sent per second.

rcv-inv/s Number of invalidation messages received from the remote server.

rcv-msg/s Total number of incoming RFS messages received per second.

dis-bread/s Number of buffer reads that would be eligible for caching if caching is not disabled. (Indicates the penalty of running uncached.)

blk-inv/s Number of buffers removed from the client cache.

EXAMPLES

The following command reports today's CPU activity so far:

```
sar
```

The following command reports CPU activity over a period of 10 minutes and saves the data:

```
sar -o temp 60 10
```

The following command reports disk and tape activity saved from a previous *sar* (like that shown above) in which data was saved:

```
sar -d -f temp
```

FILES

/usr/adm/sa/sa*dd* daily data file, where *dd* are digits representing the day of the month.

SEE ALSO

sag(1G), sar(1M).
S/Series CTIX Administrator's Guide.

NAME

sdb - symbolic debugger

SYNOPSIS

sdb [**-w**] [**-W**] [objfil [corfil [directory-list]]]

DESCRIPTION

The *sdb* command calls a symbolic debugger that can be used with C programs. It can be used to examine their object files and core files and to provide a controlled environment for their execution.

objfil is an executable program file that has been compiled with the **-g** (debug) option. If *objfil* has not been compiled with the **-g** option, the symbolic capabilities of *sdb* are limited, but the file can still be examined and the program debugged. The default for *objfil* is **a.out**. *corfil* is assumed to be a core image file produced after executing *objfil*; the default for *corfil* is **core**. The core file need not be present. A **-** in place of *corfil* forces *sdb* to ignore any core image file. The colon-separated list of directories (*directory-list*) is used to locate the source files used to build *objfil*.

It is useful to know that at any time there is a *current line* and *current file*. If *corfil* exists, they are initially set to the line and file containing the source statement at which the process terminated. Otherwise, they are set to the first line in *main()*. The current line and file can be changed by using the source file examination commands.

Initially, *sdb* has an asterisk character (*) prompt, which indicates that *sdb* is ready for the user to enter the first command. If the **S**, **s**, **I**, or **i** command is used, the prompt corresponds to the command letter (for example, **S** when the **S** command is used).

By default, warnings are provided if the source files used in producing *objfil* cannot be found, or are newer than *objfil*. This checking feature and the accompanying warnings may be disabled by the **-W** flag.

Names of variables are written just as they are in C. *sdb* does not truncate names. Variables local to a procedure may be accessed using the form *procedure*:*variable*. If no procedure name is given, the procedure containing the current line is used by default.

It is also possible to refer to structure members as *variable*.*member*, pointers to structure members as *variable*->*member*, and array elements as *variable*[*number*]. Pointers may be dereferenced by using the form *pointer*[0]. Combinations of these forms may also be used. A number may be used in place of a structure variable name. In this case, the number is viewed as the address of the structure, and the template used for the structure is that of the last

structure referenced by *sdb*. An unqualified structure variable may also be used with various commands. Generally, *sdb* will interpret a structure as a set of variables. Thus, *sdb* will display the values of all the elements of a structure when it is requested to display a structure. An exception to this interpretation occurs when displaying variable addresses. An entire structure does have an address, and it is this value *sdb* displays, not the addresses of individual elements.

Elements of a multidimensional array may be referenced as *variable [number][number]...*, or as *variable [number,number,...]*. In place of *number*, the form *number;number* may be used to indicate a range of values, * may be used to indicate all legitimate values for that subscript, or subscripts may be omitted entirely if they are the last subscripts and the full range of values is desired. As with structures, *sdb* displays all the values of an array or of the section of an array if trailing subscripts are omitted. It displays only the address of the array itself or of the section specified by the user if subscripts are omitted.

A particular instance of a variable on the stack may be referenced by using the form *procedure:variable,number*. All the variations mentioned in naming variables may be used. *Number* is the occurrence of the specified procedure on the stack, counting the top, or most current, as the first. If no procedure is specified, the procedure currently executing is used by default.

It is also possible to specify a variable by its address. All forms of integer constants that are valid in C may be used, so that addresses may be input in decimal, octal, or hexadecimal.

Line numbers in the source program are referred to as *file-name:number* or *procedure:number*. In either case, the number is relative to the beginning of the file. If no procedure or filename is given, the current file is used by default. If no number is given, the first line of the named procedure or file is used.

While a process is running under *sdb*, all addresses refer to the executing program; otherwise, they refer to *objfil* or *corfil*. An initial argument of -w permits overwriting locations in *objfil*.

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*) and the *file address* corresponding to a written *address* is calculated as follows:

b1<=*address*<*e1*

file address=*address*+*f1*-*b1*

otherwise,

b2<=*address*<*e2*

file address=*address*+*f2*-*b2*.

Otherwise, the requested *address* is not legal. In some cases (for example, for programs with separated I and D space), the two segments for a file may overlap.

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

Commands

The commands for examining data in the program are as follows:

t Prints a stack trace of the terminated or halted program.

T Prints the top line of the stack trace.

variable/*clm*

Prints the value of *variable* according to length *l* and format *m*. A numeric count *c* indicates that a region of memory, beginning at the address implied by *variable*, is to be displayed. The length specifiers are as follows:

b one byte

h two bytes (half word)

l four bytes (long word)

Legal values for *m* are as follows:

c character

d decimal

u decimal, unsigned

o	octal
x	hexadecimal
f	32-bit single precision floating point
g	64-bit double precision floating point
s	assume *variable* is a string pointer and print characters starting at the address pointed to by the variable
a	print characters starting at the variable's address; this format may not be used with register variables
p	pointer to procedure
i	disassemble machine-language instruction with addresses printed numerically and symbolically
I	disassemble machine-language instruction with addresses just printed numerically

Length specifiers are only effective with the **c**, **d**, **u**, **o**, and **x** formats. Any of the specifiers, **c**, **l**, and **m**, may be omitted. If all are omitted, *sdb* chooses a length and a format suitable for the variable's type as declared in the program. If **m** is specified, then this format is used for displaying the variable. A length specifier determines the output length of the value to be displayed, sometimes resulting in truncation. A count specifier **c** tells *sdb* to display that many units of memory, beginning at the address of *variable*. The number of bytes in one such unit of memory is determined by the length specifier **l** , or if no length is given, by the size associated with the *variable*. If a count specifier is used for the **s** or **a** command, then that many characters are printed. Otherwise, successive characters are printed until either a null byte is reached, or 128 characters are printed. The last variable may be redisplayed with the command *./*.

The *sh*(1) metacharacters * and ? may be used within procedure and variable names, providing a limited form of pattern matching. If no procedure name is given, variables local to the current procedure and global variables are matched; if a procedure name is specified, then only variables local to that procedure are matched. To match only global variables, the form *:pattern* is used.

linenumber?*lm*
variable:?*lm*

Prints the value at the address from **a.out** or I space given by *linenumber* or *variable* (procedure name), according to the format *lm*. The default format is 'i'.

variable=*lm*
linenumber=*lm*
number=*lm*

Prints the address of *variable* or *linenumber*, or the value of *number*, in the format specified by *lm*. If no format is given, then **lx** is used. The last variant of this command provides a convenient way to convert between decimal, octal, and hexadecimal.

variable!*value*

Sets *variable* to the given *value*. The value may be a number, a character constant, or a variable. The value must be well defined; expressions that produce more than one value, such as structures, are not allowed. Character constants are denoted '*character*. Numbers are viewed as integers unless a decimal point or exponent is used. In this case, they are treated as having the type double. Registers are viewed as integers. The *variable* may be an expression that indicates more than one variable, such as an array or structure name. If the address of a variable is given, it is regarded as the address of a variable of type *int*. C conventions are used in any type conversions necessary to perform the indicated assignment.

f Prints the 68881/68040 floating-point registers.

x Prints the machine registers and the current machine-language instruction.

X Prints the current machine-language instruction.

The commands for examining source files are as follows:

e *procedure*
e *file-name*
e *directory/*
e *directory file-name*

The first two forms set the current file to the file containing *procedure* or to *file-name*. The current line is set to the first line in the named procedure or file. Source files are assumed to be in *directory*. The default is the current working directory. The latter two forms change the value of *directory*. If no procedure, filename, or directory is given, the current procedure name and filename are reported.

/regular expression/

Searches forward from the current line for a line containing a string matching *regular expression* as in *ed*(1). The trailing / may be deleted.

?regular expression?
Searches backward from the current line for a line containing a string matching *regular expression* as in *ed*(1). The trailing ? may be deleted.

p Prints the current line.

z Prints the current line followed by the next nine lines. Sets the current line to the last line printed.

w Window. Prints the 10 lines around the current line.

number
Sets the current line to the given line number. Prints the new current line.

count+
Advances the current line by *count* lines. Prints the new current line.

count−
Retreats the current line by *count* lines. Prints the new current line.

The commands for controlling the execution of the source program are as follows:

count **r** *args*
count **R**
Runs the program with the given arguments. The **r** command with no arguments reuses the previous arguments to the program while the **R** command runs the program with no arguments. An argument beginning with < or > causes redirection for the standard input or output, respectively. If *count* is given, it specifies the number of breakpoints to be ignored.

linenumber **c** *count*
linenumber **C** *count*
Continues after a breakpoint or interrupt. If *count* is given, the program will stop when *count* breakpoints have been encountered. The signal that caused the program to stop is reactivated with the **C** command and ignored with the **c** command. If a line number is specified, a temporary breakpoint is placed at the line and execution is continued. This temporary breakpoint is deleted when the command finishes.

linenumber **g** *count*
Continues after a breakpoint with execution resumed at the given line. If *count* is given, it specifies the number of breakpoints to be ignored.

s *count*
S *count*

Single-steps the program through *count* lines. If no count is given, then the program is run for one line. **S** is equivalent to **s** except it steps through procedure calls.

i
I Single-steps by one machine-language instruction. The signal that caused the program to stop is reactivated with the **I** command and ignored with the **i** command.

variable$**m** *count*
address:**m** *count*

Single-steps (as with **s**) until the specified location is modified with a new value. If *count* is omitted, it is effectively infinity. *variable* must be accessible from the current procedure. Since this command is done by software, it can be very slow.

level **v**

Toggles verbose mode, for use when single-stepping with **S**, **s**, or **m**. If *level* is omitted, then just the current source file and/or subroutine name is printed when either changes. If *level* is 1 or greater, each C source line is printed before it is executed; if *level* is 2 or greater, each assembler statement is also printed. A **v** turns verbose mode off if it is on for any level.

k Kills the program being debugged.

procedure(arg1,arg2,...)
procedure(arg1,arg2,...)/*m*

Executes the named procedure with the given arguments. Arguments can be integer, character, or string constants or names of variables accessible from the current procedure. The second form causes the value returned by the procedure to be printed according to format *m*. If no format is given, it defaults to **d**. This facility is only available if the program was loaded with the -g option.

linenumber **b** *commands*

Sets a breakpoint at the given line. If a procedure name without a line number is given (for example, "proc:"), a breakpoint is placed at the first line in the procedure even if it was not compiled with the **-g** option. If no *linenumber* is given, a breakpoint is placed at the current line. If no *commands* are given, execution stops just before the breakpoint and control is returned to *sdb*. Otherwise, the *commands* are executed when the breakpoint is encountered and execution continues. Multiple

commands are specified by separating them with semicolons. If **k** is used as a command to execute at a breakpoint, control returns to *sdb*, instead of continuing execution.

B Prints a list of the currently active breakpoints.

linenumber **d**
Deletes a breakpoint at the given line. If no *linenumber* is given, then the breakpoints are deleted interactively. Each breakpoint location is printed and a line is read from the standard input. If the line begins with a **y** or **d**, then the breakpoint is deleted.

D Delete all breakpoints.

l Print the last executed line.

linenumber **a**
Announces. If *linenumber* is of the form *proc*:*number*, the command effectively does a *linenumber* **b l**. If *linenumber* is of the form *proc*:, the command effectively does a *proc*: **b T**.

Miscellaneous commands:

!*command*
The command is interpreted by *sh*(1).

new-line
Performs the previous command again.

end-of-file character
Scrolls and prints the next 10 lines of instructions, source, or data depending on which was printed last. The end-of-file character is usually CONTROL-D.

< *filename*
Reads commands from *filename* until the end of file is reached, and then continues to accept commands from standard input. When *sdb* is told to display a variable by a command in such a file, the variable name is displayed along with the value. This command may not be nested; < may not appear as a command in a file.

M Prints the address maps.

M [?/] [*] *b e f*

Records new values for the address map. The arguments ? and / specify the text and data maps, respectively. The first segment (*b1, e1, f1*) is changed unless * is specified; in which case, the second segment (*b2, e2, f2*) of the mapping is changed. If fewer than three values are given, the remaining map parameters are left unchanged.

" *string*

Prints the given string. The C escape sequences of the form *\character* are recognized, where *character* is a nonnumeric character.

q Exits the debugger.

The following commands also exist and are intended only for debugging the debugger:

V Prints the version number.
Q Prints a list of procedures and files being debugged.
Y Toggles debug output.

sdb may be instructed to monitor a given memory location and stop the program when the value at that location changes in any given way. For example:

```
> if x <= 123
```

The above example instructs *sdb* to monitor the value at location x. When the user gives the command to continue (c), *sdb* checks the value of x at every source line executed and stops the program if the given condition becomes true. Note that this construct slows the real-time execution of a program.

The syntax of the *if* command is as follows:

if Shows a list of the current data breakpoints; assigns a number to each.

if *var* Monitors the value of *var* and stops the program if the value changes. A variable name may be used for *var*, as well as a constant address. Comparisons are done as either 4-byte signed or 4-byte unsigned, depending on the data type. To perform a 1-byte or 2-byte comparison, an optional length value may accompany *var*. An example of a 2-byte comparison is

```
if x,2 = 0xff
```

if *var rel value*

Compares the value of *var* to the constant given and stops the program if the condition is true. The values of *rel* may be =, ==, <, <=, >, >=, or !=.

off *n* Disables or turns off a data breakpoint without removing it from the list.

on *n* Enables a breakpoint that was turned off.

out *n* Removes a breakpoint from the list.

Conditional breakpoints are used in a manner similar to data breakpoints, except that the user specifies a place in the program at which *sdb* should stop to check the data values. For example,

mysub:99 b if xyz = 123

The above example instructs *sdb* to check the value of xyz every time the program arrives at line 99 of subroutine *mysub*. If the condition is true, then execution stops there, as with a normal breakpoint. This type of breakpoint does not monitor the value xyz at every line of code, as the data breakpoint does.

FILES

a.out
core

SEE ALSO

cc(1), sh(1), a.out(4), core(4), syms(4).

WARNINGS

When *sdb* prints the value of an external variable for which there is no debugging information, a warning is printed before the value. The size is assumed to be **int** (integer).

Data stored in text sections are indistinguishable from functions.

Line number information in optimized functions is unreliable, and some information may be missing.

BUGS

If a procedure is called when the program is *not* stopped at a breakpoint (such as when a core image is being debugged), all variables are initialized before the procedure is started. This makes it impossible to use a procedure that formats data from a core image.

When setting a breakpoint at a procedure, *sdb* will inconsistently produce the incorrect line number. Recompiling the source program will correct this problem.

NAME

serstat - display serial port error statistics

SYOPNSIS

serstat

DESCRIPTION

The *serstat* command reports error status information about groups of serial tty ports. The command supports IOP16 ports. The command does not currently support IOP and RIOP ports. When first invoked, *serstat* finds the four ports with the largest number of total errors logged and displays the logged errors.

The command then runs in "automatic" mode, in which it scans all serial ports for any change of status. As port status changes, *serstat* updates the display to ensure that the four ports with the largest number of errors logged are displayed at all times. Ports with fewer errors logged are replaced as other ports with more errors logged are displayed. A message at the bottom of the screen indicates which port has most recently changed.

The *serstat* program can also be run in "scan" mode and "continuous" mode. In scan mode, *serstat* scans sequential groups of ports every 3 seconds and displays the errors. In continuous mode, *serstat* continues to scan and update the currently-displayed ports only.

To exit *serstat*, generate a keyboard interrupt.

Once *serstat* is running, use any of the following one-character commands:

r Redraws the screen. No mode change.

a Redraws the screen. Starts automatic mode.

m Redraws the screen with ports having the most errors. Starts automatic mode.

s Redraws the screen. Starts scan mode.

c Redraws the screen. Starts continuous mode.

The program displays data from the following status structure maintained by the serial driver in the kernel:

```
struct sererrstat {
        uint    se_ttyhog;      /* tty input hog status achieved (ttin) */
        uint    se_iflushed;    /* hogs input queues discarded (ttin) */
        uint    se_idropped;    /* input char(s) dropped (ttin, serrint) */
        uint    se_norbuf;      /* no receive buffer available (serrint) */
        uint    se_othrottle;   /* output throttled, low clists (T_HIWATER) */
        uint    se_oflushed;    /* hogs output queue discarded (ttxput) */
```

```
        uint    se_odropped;    /* output char(s) dropped (serxsend, sersend) */
        uint    se_notbuf;      /* no transmit buffer available (ttout) */
        uint    se_rxorun;      /* receiver overrun (serrint) */
        uint    se_exstat;      /* external status change (sertint) */
        uint    se_pe;          /* parity errors (serrint) */
        uint    se_frame;       /* CRC/framing error (serrint) */
};
```

All fields are incremented once per event occurrence except **se_idropped** and **se_odropped**. These two fields try to keep track of the number of characters dropped for that particular error event instead of the number of times that error event occurred. Note that 256 characters or more can be lost when the input queue is flushed, but the only record of this event is a single increment to the **se_iflushed** field.

The field **se_exstat** counts the number of external status changes occurring on a port. A break condition or change in the Carrier Detect or Clear To Send lines increments this number. These are not normally error conditions, but may be of interest.

SEE ALSO

termio(7).

NOTE

This utility is intended for diagnostic use by qualified system administrators; it is not a basic user command.

BUGS

Scan mode displays ports that might not be present in the system.

If run on a system without an IOP or IOP16, **serstat** reports a read error and terminates.

NAME

tio - tape io filter

SYNOPSIS

tio -r tape_device [**-b** blocksize]
tio -w tape_device [**-b** blocksize]

DESCRIPTION

tio reads from or writes to a tape device asynchronously, which results in high throughput tape streaming. If the **-r** option is used, *tio* reads from tape_device and writes to standard output; if the **-w** option is used, *tio* reads from standard input and writes to tape_device. The block size specified with **tio -r** must be the same as the block size specified with **tio -w** when the tape is made.

When end-of-tape is reached, *tio* prompts the user to choose between continuing (by inserting a new tape) or exiting. The user may select either the same tape device or a new tape device by pressing RETURN when the drive is ready.

tio accepts block sizes only in multiples of 256, and fills with NULLS any incomplete last block. The **-b** flag can be used to select a particular block size. The number specifying the block size can end with **k**, **b**, or **w** to specify multiplication by 1024, 512, or 2, respectively; a pair of numbers may be separated by **x** to indicate multiplication. For example, 128 (incomplete block, filled to 256 bytes), 256w (512 bytes), 512x4 (2048 bytes), 64k (64*1024 bytes), or 128b (128*512 bytes). The default is 65536 bytes.

Although *tio* has been optimized to support tape streaming, the user may get only partial streaming, depending on the archiving software and tape drives used. For example, *cpio*(1) is usually too slow for *tio*, especially when there are a lot of small files. On the contrary, the Cipher 990 caching tape drive is too fast for *tio* to stream.

For the quarter-inch cartridge tape drive, the user can expect an increase in performance of about 100 percent using *cpio*(1) and *tio*. For the half-inch drives (Cipher 880), the user can expect a 50 percent performance gain if *cpio*(1) is used, and about a 200 percent performance gain if *dd*(1M) is used.

EXAMPLES

```
find . -print | cpio -ocQ | tio -w /dev/rmt0

tio -r /dev/rmt0 | cpio -icQt

dd if=/dev/rdsk/c0d1s1 bs=10k | tio -w /dev/rmt/c1d0h
```

FILES

/dev/rmt*, /dev/rmt/*