VALID

Dear Customer:

Valid is pleased to announce the 8.0 release of SCALDsystem UNIX. The new 4.2 BSD UNIX operating system kernel is compatible with both the 68010 and 68020 CPU boards. While most of the changes are transparent to SCALDsystem users, the following enhancements will be apparent:

- Improved network communications and UNIX-to-UNIX Copy Program (uucp) support for hardwire and modem connections.
- Networked mail facilities.
- System accounting support.
- New ROM-resident monitor program (Combined Resident Monitor).
- More automated dump tape procedure.

Documentation is essential to the successful use of any product. Accordingly, Valid has provided the following documentation.

- **8.0 UNIX Changes Document**. Describes changes and enhancements to the operating system. Included with the Changes Document is a set of supplemental documents that describe individual changes or enhancements. These documents should be inserted into your SCALDsystem Manager's Reference Manual; supplemental documents that pertain to your users can be copied and distributed.
- **SCALDsystem UNIX Programmer's Manual Update**. An update package that upgrades your Revision C UNIX Programmer's Manual to Revision D; includes instructions for inserting/deleting individual ''man'' pages to reflect 4.2 BSD operating system as implemented on SCALDsystem.

Some anomalies have been discovered during our final testing and will be corrected as soon as possible. A list of the current anomalies is attached.

Your local Field Engineer will contact you in the near future to schedule the installation of 8.0 UNIX on your system and will do everything possible to minimize the impact on your operation. Thank you for your continued support.

If you have questions or suggestions as to how we might improve our service, please contact your Valid representative.

Sincerely,

[illegible]

Daniel Sahagun
V.P. Field Engineering

Valid Part Number: 900-00237

2820 Orchard Parkway
San Jose, CA 95134
408/945-9400
Telex 371 9004

RELEASE 8.0 UNIX™ CHANGES DOCUMENT

Manual Number: 900-00229 Rev A

10 January 1986

Valid Logic Systems Incorporated
2820 Orchard Parkway
San Jose, CA 95134
(408)945-9400 Telex 371 9004

PREFACE

OBJECTIVE

The objective of the initial 8.0 release of UNIX 4.2 BSD is to provide a stable and maintainable operating system base for VALID's SCALDsystem products and to support the new 68020-based CPU board.

OVERVIEW

The 8.0 release is the first release of the 4.2 BSD operating system. This document describes the initial 8.0 implementation in some detail and focuses on the differences and enhancements from the current 7.27 release (Valid's release of the 4.1c BSD operating system). The initial 8.0 implementation provides little new system software functionality. The intent is to first provide a 4.2 based system that is as solid as the current 4.1c based system (7.27) and then to introduce enhancements at a later date. The kernel itself is changed to 4.2, and all of the current devices are supported. The 8.0 release is available for all SCALDsystems with either a 68010 or 68020 CPU (except SCALDsystem IV which is to be updated in the near future). Utilities have been maintained at a level at least equivalent to Release 7.27. In some instances, more recent 4.2 style versions are used, but a more complete set of 4.2 utilities will not be addressed until subsequent releases.

The remainder of this document is divided into four major sections that describe differences between 4.2 UNIX and 4.1c UNIX, communications enhancements, printing/spooling enhancements, and installation considerations. Throughout these sections, references to more detailed supporting documents that are part of this documentation package are also included; an update package for the SCALDsystem UNIX Programmer's Manual is also part of the documentation package and includes man pages for all new and modified commands.

SECTION 1
SYSTEM DIFFERENCES

1.1. KERNEL CHANGES

The changes described in the following subsections have been implemented in the 4.2 kernel. A list of the supplemental documents associated with system differences follows. These documents should be included in the *SCALDsystem Manager's Reference Manual.*

- *System Call Interface* – describes differences in the system call interface between 4.1c and 4.2 BSD UNIX as implemented on the SCALDsystem.
- *How CRM Differs From VRM* – describes differences between the new ROM-resident monitor (Combined Resident Monitor) and VRM (the Valid Resident Monitor).
- *Combined Resident Monitor User's Manual* – describes Valid's new ROM-resident monitor.
- *SVS Pascal and FORTRAN Version 2.4 Release Notes* – describes changes to the optional SVS Pascal and FORTRAN compilers.

1.1.1. System Call Interface

There are several differences between the system call interface in 4.1c and 4.2 BSD UNIX as implemented on the SCALDsystem. These differences are important to users writing their own programs on the SCALDsystem and are discussed in the *System Call Interface* supplemental document included with this change package.

1.1.2. Device Support

The initial 8.0 release supports all the devices available under 4.1c/7.27. Many of the device drivers are a direct carryover from 4.1c while several have been rewritten for improved maintainability or performance. The device drivers that are different in 4.2 are the console UART, the Rimfire disk/tape, and the octal serial communications (UNIX terminal) board. Console and serial port operations appear unchanged although the drivers have changed.

A new Rimfire 44/45 driver has been written for 4.2. This driver has been significantly cleaned up from a maintenance standpoint and provides the same functions as the old one. It is transparent to the user in most respects. The external differences are limited to a device name change for the disk, and the integration of the *dk*-style (so named for the *dk* disk ioctl utility; a man page for *dk*(8) is included in the update package for the SCALDsystem UNIX Programmer's Manual) ioctls that previously applied only to the Interphase SMD driver.

The Rimfire disk may now be referred to as *rd.* This is not something the user needs to worry about because the old *rim*-style naming is still supported for backward compatability. It is useful to know because the user may see some error messages printed on the console that originate in the driver and contain the name "rd." The fact that such messages refer to the disk/tape subsystem is important information. The practical result of all this is that you refer to the disk as *rim* as you always have and everything continues to work as you would expect.

The old *rimioctl* ioctls have been replaced by the *dk* ioctls. This means that the *rimioctl* command no longer is found on the system.

The old ioctls supported a number of Rimfire-specific operations such as retensioning a cartridge tape, turning on and off the Rimfire's "bus lock" mode, and maintaining a bad block list retained by the driver. These are no longer supported as they are unnecessary with the new driver.

The new *dk*-style ioctls work for both the *is* and *rd* drivers. This is a significant improvement as the previous set of ioctls were controller-type dependent, which made it difficult to write utilities for both drive types. A manual page explaining the use of *dk*(8) is included in the update package for the SCALDsystem UNIX Programmer's Manual.

The various tape sensitivity fixes that were incorporated in 7.27 have been included in the new rimfire driver along with other bug fixes.

The SMD disk driver (the *is* driver) is included and supports the Interphase 2190 controller and Fujitsu Eagle disk drive as it did in 4.1c/7.27. The Eagle disk is referred to as *id.*

In subsequent releases, Valid will introduce support for the Storager, a new Interphase controller that handles ST506 and ESDI 5-1/4 inch disks and 1/4 inch tape. The *is* driver is upgraded to control both the 2190 SMD controller and the Storager ST506/EDSI controller. An *it* driver will be introduced to control the Storager 1/4 inch tape drive. This support will allow an increase in performance and will decrease the size of the disk sub-system. The Storager controls up to two drives; the initial capacity of a single drive is expected to range from 170 Mbytes up to 380 Mbytes.

Valid will introduce support for a smart octal serial board capable of running at 9600 baud in future upgrades to the 8.0 system software release.

1.1.3. Memory Map and Dump Tapes

A complete memory map of the physical address space supported by the kernel in its S-32 configuration is shown in table 1-1. The memory map is useful for strapping boards and for correctly dumping memory for a crash tape. The proper procedure for making a crash tape as been simplified under the new Combined Resident Monitor that supports a command called COREDUMP to automate this process. Should your system crash, simply load a tape in the default tape drive and type:

```
COREDUMP
```

If you have a 1/2-inch drive, leave the tape in the drive as it is positioned; if you have a 1/4-inch drive, remove the cartridge, label it clearly as a "Core Image Dump Tape" and insert a new cartridge to hold the *tar*ed copy of vmunix (label this tape "tar vmunix").

Boot vmunix up in single-user mode and tar vmunix onto the tape by typing:

```
tar c vmunix
```

Fill out a Software Problem Report and return the tape(s) to the Valid Software Release group.

Table 1-1. Physical Memory Map

Device	Unit	Base	Size	Limit
Memory		000000	380000	37FFFF
3Com	ec0	380000	2000	381FFF
3Com	ec1	382000	2000	383FFF
3Com	ec2	384000	2000	385FFF
3Com	ec3	386000	2000	387FFF
3Com	ec4	388000	2000	389FFF
3Com	ec5	38A000	2000	38BFFF
3Com	ec6	38C000	2000	38DFFF
3Com	ec7	38E000	2000	38FFFF
CCB		390000	10000	39FFFF
VGB	vg1	3A0000	10000	3AFFFF
VGB	vg2	3B0000	10000	3BFFFF
VGB	vg3	3C0000	10000	3CFFFF
PIB	vp0	3D0000	10000	3DFFFF
BAB		3E0000	10000	3EFFFF
VGB	vg0	3F0000	10000	3FFFFF
Memory		400000	400000	7FFFFF
CRB		800000	100000	8FFFFF
Memory		900000	480000	D7FFFF
unused		D80000	400000	DBFFFF
Realchip	2	DC0000	20000	DDFFFF
Realchip	3	DE0000	20000	DFFFFF
CGB*	0	E00000	40000	E3FFFF
CGB	1	E40000	40000	E7FFFF
CGB	2	E80000	40000	EBFFFF
CGB	3	EC0000	40000	EFFFFF
GAB	0	E80000	20000	E9FFFF
GAB	1	EA0000	20000	EBFFFF
GAB	2	EC0000	20000	EDFFFF
GAB	3	EE0000	20000	EFFFFF
Realchip	0	F00000	20000	F1FFFF
Realchip	1	F20000	20000	F3FFFF
unused		F40000	90000	FCFFFF
Realfast		FD0000	30000	FDFFFF
unused		FE0000	10000	FEFFFF
User Alloc		FF0000	10000	FFFFFF

*Memory space from E80000 to EFFFFF may be devoted to a combination of non-overlapping CGBs and GABs; all numbers are hexadecimal.

1.1.4. Graphics

Support for graphics and hardcopy devices has been changed very little between 4.1c/7.27 and the initial 8.0 release. Color and monochrome video graphics boards are supported using the board code and device drivers from 7.27. All *loadboard* procedures continue to operate as in the past release. The hardcopy and communications activities of the PIB also are supported with the same board code and drivers as in 7.27. Accordingly, all display manager, hardcopy, DR11, and RSCS functions are unchanged.

There is one program in the display subsystem that has been rewritten. It is the window manager program. The new window manager provides the same functions as the old program, but overcomes the numerous process control problems associated with 7.27 windows. These problems include "undead processes associated with this window" and other similar problems. If there are window manager problems, please refer to the following procedure.

1. **Kill All Window Managers**

   From the maintenance console, enter **ps aux** and note the process ID numbers of all processes called *wm*. Use *kill*(1) to terminate these processes (e.g., **kill -9** *ID number*).

2. **Log Out of All Windows**

   Log out of all active windows and delete the default window. If this is not possible, delete the */etc/utmp* file.

3. **Load the VG Boards**

   Enter the following commands from the maintenance console to reload the VG boards:

   > **/etc/loaddmgr2 > /dev/dmgrload < /lib/dmgr2.ld**
   > **/etc/loadboard -r** *number_of_VGs* **/dev/vgload0 /lib/vgin2**

   Note that the */etc/loadboard* command takes the number of VG boards in the system as an argument.

4. **Restart the Window Manager**

   From the maintenance console, restart the window manager for each VG board by entering

   > **/etc/wm** *VG_board_number* **1 &**

   for each VG board (replace *VG_board_number* with the number of the VG board; e.g., 0 for the first VG board, 1 for the second, etc.). Note that the apersand ('&') at the end of the command line **must** be entered to place the job in the background.

1.2. BOOT CODE CHANGES

The boot code is a small disk-resident program that the ROM-resident monitor loads and executes. The boot code, in turn, loads and executes a "bootable" program (i.e., vmunix) from the disk. A new version of the boot code, which adds support for the Interphase controllers and some added functionality, is included in this release.

The boot code prompts with a message that states the processor type and expects a file system/file name argument. This argument is of the form *xx*(*y*,*z*)*filename* where *xx* is the name of the controller type from which to boot (currently **rd** for Rimfire disk or **id** for Interphase disk (note that **rim** is still available as a synonym for **rd**). The *y* argument is the unit number which is a combination of the controller number and the drive number; the controller number is in the high byte, and the drive number is in the low byte (the controller number is almost always 0 so that the unit number is effectively the drive number). The *z* argument is the disk partition number and is usually the root partition (partition 6 on both the Rimfire and Interphase disks). The *filename* argument is the UNIX pathname of the file to be booted and is almost always *vmunix*.

For example, the boot code argument **rd(0,6)vmunix** refers to the file *vmunix* located on the sixth partition (normally the root partition) of the zeroth disk on the zeroth Rimfire controller, and the boot code argument **id(0,8)vmunix.new** refers to the file *vmunix.new* located on the eighth partition of the zeroth disk on the zeroth Interpahse controller.

Two abbreviated sequences are supported to allow UNIX to be booted with a minimum of keystrokes.

- If just a RETURN is entered at the prompt, vmunix is booted from the root partition on the default drive.
- If "*.suffix*" is entered, the boot code substitutes the string *x***d(0,6)vmunix.***suffix* where *x* is the letter corresponding to the default drive. This sequence is useful when an alternate kernel (e.g., *vmunix.bak*) is being booted.

1.3. LIBRARY CHANGES

This section describes differences in the C library interface between 4.1c and 4.2 BSD UNIX as implemented on the SCALDsystem. There are very few incompatabilities in the libraries aside from changes due to differences in the system call interface (see the supplemental *System Call Interface* document).

The following information was extracted from the Berkeley **Bug Fixes and Changes in 4.2BSD** document.

stdio	End-of-file marks now "stick." That is, all input requests on a stdio channel after encountering end-of-file return end-of-file until a *clearerr* call is made. This has implications for programs that use *stdio*(3S) to read from a terminal and do not process end-of-file as a terminating keystroke.
getwd	Has been moved from the old jobs library to the standard C run-time library. *Getwd*(3) now returns an error string rather than printing on the standard error when unable to decipher the current working directory.
perror	Now uses the writev system call to pass all its arguments to the system in a single system call. This has profound effects on programs that transmit error messages across a network.

setjmp And longjmp now save and restore the signal mask so that non-local exit from a signal handler is transparent. The old semantics are available with _setjmp and _longjmp.

The only other problem is the lack of a *locv*(3) entry. On the 68000 family, a long is the same as an integer, so *iocv*(3) may be used in it's place.

1.4. HEADER FILES

All standard header files are now found in */usr/include* instead of */include* under 7.27. This move has been done to provide a system that more closely resembles a standard 4.2 UNIX system.

The include files are all derived from Berkeley 4.2 except the following that have been carried over from 7.27:

a.out.h
dumprestor.h
dklabel.h
fstab.h

The following include files have been removed:

autoconf.h
bootstrap.h
whoami.h
descrip.h
uparm.h

All programs should be recompiled using the new include files and new *libc.a*. One should be especially careful about programs that update/read the *utmp* or *wtmp* files since the utmp structure has changed. The stat structure defined in */usr/include/sys/stat.h* has the extra field "st_blocks" defined, but this shouldn't affect any user programs. The names of the flags used by the system call *access*(2) have been changed in */usr/include/sys/file.h*. The following table shows the old flag along with it's new counterpart. See *access*(2) in the SCALDsystem UNIX Programmer's manual for more information.

FACCESS_EXISTS	→ F_OK
FACCESS_EXECUTE	→ X_OK
FACCESS_WRITE	→ W_OK
FACCESS_READ	→ R_OK

Almost all header files are upward compatible from 7.27; the only one that should give users trouble is <*sys/file.h*> as mentioned before.

1.5. LANGUAGES DIFFERENCES

C, FORTRAN 77, and Pascal are supported under the 4.2 system. The C compiler is the same MIT-based compiler currently shipped with the 7.27 systems. The FORTRAN 77 and Pascal compilers are the latest available from SVS (Version 2.4). All three compilers are source code compatible with the previous versions, although there are new features available in the SVS products.

NOTE

> ALL Pascal and FORTRAN 77 programs must be recompiled with Version 2.4 of the SVS compilers.

There have been changes made to the object file format and the calling sequences. Modules compiled with the new compilers cannot be link with modules compiled with previous versions.

1.5.1. C Compiler

There are no changes to the C compiler or to the language accepted by the compiler.

A number of changes have been made to the C compiler executive *cc*(1). The **-z** flag, which causes pacifier messages to be printed on the terminal, has been renamed **-Z** to avoid conflict with the Berkeley loader's **-z** flag, which specifies that 413 (demand-loaded) binaries are to be created (see the *ld* discussion below). The **-sp** flag, which disables the generation of stack probe instructions, is now the default (stack probing is unnecessary on the 68010; it is required on the 68000 to prevent system crashes). You can get stack probe instructions inserted into your compiled code by specifying the **-sP** flag. Finally, the executive looks in */lib* for some of the passes of the compiler that used to be in /bin (specifically, ccom and c2).

1.5.2. Assembler

The UNIX assembler exists entirely to support the C compiler. It has been changed in two ways. First, it now expects its standard assembler header to be in */usr/include/as.h* which was previously located in */include/as.h*. Second, it treats lines that begin with a '#' as comments to allow use of the C preprocessor that emits such comment lines on assembly language files.

1.5.3. Pascal

There are be no changes to the Pascal compiler, or to the language accepted by the compiler.

1.5.4. FORTRAN

SVS FORTRAN 77 Version 2.4 includes several new features that all may be characterized as extensions to the ANSI standard and the language accepted by earlier versions of SVS FORTRAN. These features should not cause any existing programs to fail.

Bitwise Operations on Integers

DATA Initialization of Named COMMON Blocks

Different Sized COMMON Blocks Allowed

Argument Type Mismatch Allowed

Further, there is a change in the way that character data types are passed as parameters. They are now passed as a 4-byte pointer followed by a 4-byte length filed. This change was made for compatability with C, as C has no parameter data type corresponding to a 2-byte value.

For a more detailed description of SVS Fortran 77 changes, see the *SVS Version 2.4 Release Notes* supplemental document.

1.5.5. The Linker

The *ld* linker defaults to demand-paged (413) loading. Use the **-N** flag to turn off demand-paged loading for programs that are not to be loaded demand-paged.

1.6. UTILITIES DIFFERENCES

The initial release of the 8.0 system supports all of the utilities, at the same level of functionality, as the 7.27 system, with few exceptions. */etc/rimioctl*, */lib/libjobs.a*, */lib/libnet.a*, and */bin/dcheck* have been removed because they are obsolete. Other utilities, such as *clri* have been moved from */bin* to */etc*. In other cases, the functionality has been upgraded to meet the 4.2 semantics, or the 4.2 source has been brought up, instead of the 7.27 version.

The following utilities have been removed from the initial 8.0 release:

rimioctl	- obsoleted by dk(8)
setmnt	- obsoleted by mount(8)
restor	- obsoleted by restore(8)
loaddmgr	- replaced by loaddmgr2

The following utilites have been added to the initial 4.2:

ci (1)	- rcs source control system utility
co (1)	- rcs source control system utility
dk (8)	- disk control program
dkbn (8)	- disk blocknumber utility
dklabel (8)	- disk label program
dktest (8)	- disk test program
eqn (1)	- equation preprocessor for troff(1)
expand, unexpand (1)	- tabs to spaces, and back
finger (1)	- user information lookup program
ftp (1C)	- file transfer program
ident (1)	- rcs source control system utility
libdbm.a (3)	- data base library for newaliases(1)
lock (1)	- reserve a terminal
m4 (1)	- macro processing package
Mail (1)	- Berkeley 4.2 mail program
makewhatis (8)	- rebuild database used by man(1)
merge (1)	- three-way file merge
msgs (1)	- system messages and junk mail program
mt (1)	- magnetic tape manipulating program
newaliases (1)	- rebuild mail(1) alias data base
praliases (1)	- print aliases in mail data base
prmail (1)	- print out mail
rcp (1C)	- remote file copy
rcs (1)	- rcs source control system utility
rcsdiff (1)	- rcs source control system utility
rcsmerge (1)	- rcs source control system utility
reset (1)	- reset the teletype bits
restore (8)	- file system restore (replaces restor(8))
rlog (1)	- print RCS log messages, info
rmail (1)	- handle remote mail received via uucp
rvtroff (1)	- rotated vtroff output
sa (8)	- system accounting
sccstorcs (1)	- rcs source control system utility
sccstorcs (8)	- build RCS file from SCCS file
sendmail (8)	- send mail over the internet
talk (1)	- talk to another user
tbl (1)	- table preprocessor for troff(1)
telnet (1C)	- user interface to TELNET protocol
tip (1)	- connect to a remote system
uuclean (8C)	- uucp spool directory clean-up
uucp (8)	- uucp utility

uudecode (8)	- uucp utility
uuencode (8)	- uucp utility
uulog (8)	- uucp utility
uuname (8C)	- return list of remote uucp sites
uupoll (8C)	- poll remote uucp sites
uuq (8C)	- report uucp files spooled per site
uurate (8C)	- report uucp transfer rate
uusend (8)	- uucp utility
uusnap (8C)	- show snapshot of UUCP system
uuusage (8C)	- report uucp usage by user name
uux (8)	- uucp utility
vmstat (1)	- report virtual memory statistics
whatis (1)	- describe what a command is
whereis (1)	- locate program source, binary, or manual
which (1)	- locate a program file

The following utilities have been replaced by their 4.2 versions or have been otherwise modified:

ctags (1)	- recognizes fortran files now
egrep (1)	- now handles multiple files correctly
getty (8)	- supports new gettytab speed table
init (8)	- supports new signal handler
iostat (1)	- works better than it did
install (8)	- more options
last (1)	- shows remote name
libcurses.a (3)	- various bug fixes
login (1)	- various bug fixes
lpt (1)	- 4.2 version
man (1)	- now supports "whatis" database
more (1)	- various bug fixes
mkfs, newfs (8)	- user can specify inode density
nm (1)	- works on libraries, too
od (1)	- more options
pstat (1)	- needed for 4.2 kernel
rc (8)	- calls rc.network to handle network daemons
tar (1)	- more options
umount (8)	- uses umount syscall instead of unmount

1.6.1. Location of Utilities

Many of the utilities have been moved to make the root file system appear more like a standard 4.2 UNIX system. Most of the UC-Berkeley programs have been moved to */usr/ucb* as in 4.2; likewise, many Valid-specific utilities have migrated to */usr/valid/bin*. The intent of this reorganization is to present a file structure that is familiar to UNIX users and more closely matches the supplied documentation. A beneficial side effect is that command searches of crowded directories like */bin* can be made more quickly; this benefit is most evident to C shell users who employ pathname hashing to find commands.

Two new entries are in the standard path; */usr/ucb* and */usr/valid/bin*. The directory structures on the following pages shown the new utility directories on the 8.0 system.

The following list of files comprises the 8.0 Release:

```
root:
----

.cshrc      bin       lib           special.cpio  u0
.login      dev       lost+found    tmp           usr
.profile    etc       net           vmunix
```

```
bin:
---

Vpq         csh         hpr         pr              tar
Vpr         date        hprm        ps              tee
Vprm        dd          kill        pwd             test
[           df          ld          rm              time
adb         diff        ln          rmail           tp
ar          du          login       rmdir           true
as          echo        ls          rvcat           tty
awk         ed          mail        rvsort          vcat
binmail     expr        make        rvtroff         vpl
cat         false       mkdir       rvtroff.valid   vpq
cc          filecopy    mt          sed             vpr
chgrp       find        mv          setup           vprm
chmod       ged         newgrp      setup.small     vsort
cmp         getline     nice        sh              vtroff
conn        hostid      nm          stty            vtroff.valid
cp          hostname    od          su              who
cpio        hpq         passwd      sync            write
```

```
dev:
---

RSCS        is1a    mt1600      ris04   ris32       swap    tty38
RSCS0       is1b    mt1600nr    ris05   ris33       tty     tty39
RSCS1       is1c    null        ris06   ris34       tty00   tty3a
arp         is1d    ormt0       ris07   ris35       tty01   tty3b
cgb0        is1e    ormt1       ris08   ris36       tty02   ttya
cgb1        is1f    ptyp0       ris09   ris37       tty03   ttyb
cgb2        is20    ptyp1       ris0a   ris38       tty04   ttyc
cgb3        is21    ptyp2       ris0b   ris39       tty05   ttyd
colorctl    is22    ptyp3       ris0c   ris3a       tty06   ttye
colormem    is23    ptyp4       ris0d   ris3b       tty07   ttyf
console     is24    ptyp5       ris0e   ris3c       tty08   ttyg
ct0         is25    ptyp6       ris0f   ris3d       tty09   ttyh
dmgrload    is26    ptyp7       ris10   ris3e       tty0a   ttyp0
drum        is27    ptyp8       ris11   ris3f       tty0b   ttyp1
enet10      is28    ptyp9       ris12   rmt0        tty10   ttyp2
ffp         is29    rct0        ris13   rmt1        tty11   ttyp3
gab0        is2a    realchip0   ris14   rmt1600     tty12   ttyp4
gab1        is2b    realchip1   ris15   rmt1600nr   tty13   ttyp5
gab2        is2c    realchip2   ris16   rrim00      tty14   ttyp6
gab3        is2d    realchip3   ris17   rrim06      tty15   ttyp7
is00        is2e    realfast    ris18   rrim08      tty16   ttyp8
is01        is2f    rim00       ris19   rrim09      tty17   ttyp9
is02        is30    rim06       ris1a   rrim0a      tty18   vgcom0
is03        is31    rim08       ris1b   rrim0b      tty19   vgcom1
is04        is32    rim09       ris1c   rrim10      tty1a   vgcom2
is05        is33    rim0a       ris1d   rrim11      tty1b   vgcom3
is06        is34    rim0b       ris1e   rrim20      tty20   vgdpy0
is07        is35    rim10       ris1f   rrim21      tty21   vgdpy1
```

```
is08       is36       rim11      ris20      rrim30     tty22      vgdpy2
is09       is37       rim20      ris21      rrim31     tty23      vgdpy3
is0a       is38       rim21      ris22      rrim40     tty24      vghcm
is0b       is39       rim30      ris23      rrim41     tty25      vgload0
is0c       is3a       rim31      ris24      rrim50     tty26      vgload1
is0d       is3b       rim40      ris25      rrim51     tty27      vgload2
is0e       is3c       rim41      ris26      rrim60     tty28      vgload3
is0f       is3d       rim50      ris27      rrim61     tty29      vms
is10       is3e       rim51      ris28      rrim70     tty2a      vp0
is11       is3f       rim60      ris29      rrim71     tty2b      vpcom
is12       kmem       rim61      ris2a      smoct0     tty30      vpload
is13       lasar      rim70      ris2b      smoct1     tty31      vpload0
is14       log        rim71      ris2c      smoct2     tty32      vplot
is15       lp         rimboot0   ris2d      smoct3     tty33      vprint
is16       maint      ris00      ris2e      smoct4     tty34      xcons
is17       mem        ris01      ris2f      smoct5     tty35
is18       mt0        ris02      ris30      smoct6     tty36
is19       mt1        ris03      ris31      smoct7     tty37

etc:
---

ac               ether            mkfs             remote           smdwl
accton           filesize         mklost+found     restore          talkd
arp              fsck             mknewpwentry     rexecd           tcpdiscardd
backup           ftpd             mknod            rlogind          tcpsystatd
backup.2         getfs            mkproto          route            telnetd
backup.3         getty            mkscaldusr       routed           termcap
chown            gettytab         mktimezone       rpcserver        tftpd
clri             graphicsbanner   mkusr            rshd             tunefs
conf.default     group            mount            sa               udpdiscardd
cron             halt             ncheck           services         umount
dcheck           icheck           networks         setdate          update
debug            ifconfig         newfs            shutdown         uucp.daily
disktab          ind              omni             shutdown.local   uucp.hourly
dk               indaemons        passwd           smdfo            uurate
dkbn             init             phones           smdfvclear       uuusage
dkconfig         install          printcap.proto   smdfvset         vmstat
dklabel          loadboard        protocols        smdhelp          wall
dktest           loaddmgr2        pstat            smdmba           wm
downacts         loadskyffp       rc               smdmcf           wmstarter
dump             lpc              rc.network       smdnd
efsioctl         mkconfig         reboot           smdstat

u0:
--

doc          layout       man          scald        tmp          usr.valid
editor       library      msgs         spool        user         vfont
fortran      lost+found   pascal       src          usr.bin

u0/editor:
---------

MakeAddToList  lib            recover        startup.ged
doc            make           softkeyassign  update

u0/editor/doc:
-------------

add          display      help.wrk     reattach     smash
assign       dot          ignore       redo         spin
```

```
auto           edit          library       remove        split
backannotate   error         lineedit      replace       swap
bubble         exit          mirror        return        undo
change         filenote      move          rotate        update
check          find          next          scale         use
circle         format        note          script        vectorize
copy           get           paint         section       version
cut            grid          paste         set           window
delete         group         pinswap       show          wire
diagram        hardcopy      property      signame       write
directory      help          quit          simulate

u0/editor/lib:
-------------

ged.ex    hpfilter  plottime

u0/editor/make:
--------------

MakeNewDrawings   makeanyway        makekeepbinaries  whouses.edl
MakeNoBins.sh     makefile          whouses
looking.ed        makejustlooking   whouses.ed

u0/fortran:
----------

code             fortran         library.o       sky.paslib.obj  wraplib.o
dbg              ftnlib.obj      paslib.obj      ulinker

u0/layout:
---------

compare.ex          hpsim.cmap          readgds2            stipples
dac                 led2stream.ex       rotate              stipples.color
drc.ex              plot.ex             spice.ex.hardware   stream2led.ex
hpsim               plot.ex.color       spice.ex.software   writegds2

u0/library:
----------

library

u0/pascal:
---------

code             pascal          shared          ulinker
dbg              paslib.obj      sky.paslib.obj  wraplib.o

u0/pascal/shared:
----------------

cwrap.o          fakeged         paslib.obj      simasmio.o      unixtime.obj
escseq.o         fopen.obj       patch.o         sky.paslib.obj  wrap.o
escseq.obj       mon.o           rangechk        subp.o          wraplib.o
fakedir.o        msmap.o         simasm.o        subp.obj

u0/scald:
```

```
cmperrors.mem  dial          linker         property.dat   simulator
compiler       interface     packager       section        textmacro.dat
config.dat     lcshorten     plottalk       shrtassign     verifier

u0/scald/compiler:
-----------------

comp           comperr       pcomp          usrassign.sh  xefss

u0/scald/dial:
-------------

INDEX          dial.obj      procs.pas      usermakefile   userunit.pas
consts.pas     dialassign    template.pas   userunit.make  vars.pas
dial.crf       dialint.obj   types.pas      userunit.obj

u0/scald/interface:
------------------

bom.assign      glib           gspares          scald.assign
fileassign      gphysdly       lib.assign       spares.assign
gbom            gscald         physdly.assign

u0/scald/linker:
---------------

check           libet.a           link             lnkcomp
checket         libet80parser.a   linker           new4.o
heapfix         libetxface.a      lnkassign.sh

u0/scald/packager:
-----------------

packager   pckassign  pckerrdoc

u0/scald/plottalk:
-----------------

arc.o           debug.o           hpf.o            plottalk.c
attr.o          doc               imagen.o         plottalk.h
binary.o        expandpibcode.o   line.o           plottalk.o
body.o          font.c            makefile         pset.o
calcomp.o       font.o            malloc.o         readgedfile.o
char.o          get.o             openoutput.o     transform.o
circle.o        getinfo.o         otherfonts.c     trig.o
cls.o           hp.o              otherfonts.o     utils.o
color.h         hpf.c             pib.o

u0/scald/section:
----------------

secassign.sh  section

u0/scald/simulator:
------------------
```

```
alumem.int      mkucpsim        rfastimp.obj    sim             version.obj
clear.o         monitor.int     rfmap.o         sim.obj
duck.o          msmap.o         runbatchsim     simproper.obj
inoutimp.obj    rchipimp.obj    sgutsimp.obj    switchchar.o
miscsim.obj     rchipmem.o      shioimp.obj     userglob.obj


u0/scald/verifier:
------------------

verifier


u0/spool:
---------

B9424       Vpd22       at            hp7475      lpd         uucp
Vpd         Vpd36       calcomp1043   hp7580      mail        uucppublic
Vpd11       Vpd42       comp5744      hpd         mqueue      vpd


u0/spool/at:
------------

past


u0/spool/uucp:
--------------

OLD   XTMP


u0/src:
-------

com232.vax


u0/user:
--------

370         ALL         STANDALONE  VAX


u0/user/370:
------------

filecopy.cmd


u0/user/ALL:
------------

.cshrc        .profile        packager.cmd  startup.ged   verifier.cmd
.login        compiler.cmd    simulate.cmd  td.cmd


u0/user/VAX:
------------

filecopy.cmd


u0/vfont:
---------
```

```
B.10           R.14r          bodoni.i.10r      fix.14            oldenglish.8r
B.10r          R.16           bodoni.r.10       fix.14r           pip.16
B.11           R.16r          bodoni.r.10r      fix.6             pip.16r
B.11r          R.18           chess.18          fix.6r            playbill.10
B.12           R.18r          chess.18r         fix.9             playbill.10r
B.12r          R.20           clarendon.14      fix.9r            railmag
B.14           R.20r          clarendon.14r     gacham.b.10       script.18
B.14r          R.22           clarendon.18      gacham.b.10r      script.18r
B.16           R.22r          clarendon.18r     gacham.i.10       shadow.16
B.16r          R.24           cm.b.10           gacham.i.10r      shadow.16r
B.18           R.24r          cm.b.10r          gacham.r.10       sign.22
B.18r          R.28           cm.b.11           gacham.r.10r      sign.22r
B.20           R.28r          cm.b.11r          graphics.14       st.b.10r
B.20r          R.36           cm.b.12           graphics.14r      st.i.10r
B.22           R.36r          cm.b.12r          greek.10          st.r.10r
B.22r          R.6            cm.b.6            greek.10r         stare.b.10
B.24           R.6r           cm.b.6r           h19.10            stare.b.10r
B.24r          R.7            cm.b.7            h19.10r           stare.b.11
B.28           R.7r           cm.b.7r           hebrew.16         stare.b.11r
B.28r          R.8            cm.b.8            hebrew.16r        stare.b.12
B.36           R.8r           cm.b.8r           hebrew.17         stare.b.12r
B.36r          R.9            cm.b.9            hebrew.17r        stare.b.14
B.6            R.9r           cm.b.9r           hebrew.24         stare.b.14r
B.6r           S.10           cm.i.10           hebrew.24r        stare.b.16
B.7            S.10r          cm.i.10r          hebrew.36         stare.b.16r
B.7r           S.11           cm.i.11           hebrew.36r        stare.b.8
B.8            S.11r          cm.i.11r          instructions      stare.b.8r
B.8r           S.12           cm.i.12           meteor.b.10       stare.b.9
B.9            S.12r          cm.i.12r          meteor.b.10r      stare.b.9r
B.9r           S.14           cm.i.6            meteor.b.12       stare.i.10
I.10           S.14r          cm.i.6r           meteor.b.12r      stare.i.10r
I.10r          S.16           cm.i.7            meteor.b.8        stare.i.11
I.11           S.16r          cm.i.7r           meteor.b.8r       stare.i.11r
I.11r          S.18           cm.i.8            meteor.i.10       stare.i.12
I.12           S.18r          cm.i.8r           meteor.i.10r      stare.i.12r
I.12r          S.20           cm.i.9            meteor.i.8        stare.i.14
I.14           S.20r          cm.i.9r           meteor.i.8r       stare.i.14r
I.14r          S.22           cm.r.10           meteor.r.10       stare.i.16
I.16           S.22r          cm.r.10r          meteor.r.10r      stare.i.16r
I.16r          S.24           cm.r.11           meteor.r.12       stare.i.8
I.18           S.24r          cm.r.11r          meteor.r.12r      stare.i.8r
I.18r          S.28           cm.r.12           meteor.r.8        stare.i.9
I.20           S.28r          cm.r.12r          meteor.r.8r       stare.i.9r
I.20r          S.36           cm.r.6            mona.24           stare.r.10
I.22           S.36r          cm.r.6r           mona.24r          stare.r.10r
I.22r          S.6            cm.r.7            nonie.b.10        stare.r.11
I.24           S.6r           cm.r.7r           nonie.b.10r       stare.r.11r
I.24r          S.7            cm.r.8            nonie.b.12        stare.r.12
I.28           S.7r           cm.r.8r           nonie.b.12r       stare.r.12r
I.28r          S.8            cm.r.9            nonie.b.8         stare.r.14
I.36           S.8r           cm.r.9r           nonie.b.8r        stare.r.14r
I.36r          S.9            countdown.22      nonie.i.10        stare.r.16
I.6            S.9r           countdown.22r     nonie.i.10r       stare.r.16r
I.6r           apl.10         cyr.o             nonie.i.12        stare.r.8
I.7            apl.10r        cyrillic.12       nonie.i.12r       stare.r.8r
I.7r           basker.b.12    cyrillic.12.r     nonie.i.8         stare.r.9
I.8            basker.b.12r   cyrillic.12r      nonie.i.8r        stare.r.9r
I.8r           basker.i.12    delegate.b.12     nonie.r.10        times.b.10
I.9            basker.i.12r   delegate.b.12.    nonie.r.10r       times.b.10r
I.9r           basker.r.12    delegate.b.12r    nonie.r.12        times.i.10
I.sr.36        basker.r.12r   delegate.i.12     nonie.r.12r       times.i.10r
R.10           bocklin.14     delegate.i.12r    nonie.r.8         times.r.10
R.10r          bocklin.14r    delegate.r.12     nonie.r.8r        times.r.10r
R.11           bocklin.28     delegate.r.12r    oldenglish.14     times.s.10
R.11r          bocklin.28r    fix.10            oldenglish.14r    times.s.10r
```

```
R.12             bodoni.b.10      fix.10r          oldenglish.18   ugramma.10
R.12r            bodoni.b.10r     fix.12           oldenglish.18r  ugramma.10r
R.14             bodoni.i.10      fix.12r          oldenglish.8


usr:
---

bin       doc       lib       msgs      preserve  src       valid
dict      include   man       news      spool     ucb


usr/bin:
-------

Fortran          diff3            lorder           rmlink           tbl
HFPfortran       drc              m4               rsend            tip
admin            egrep            makechipsfiles   runinterface     touch
at               eqn              makedrawingnam   ruptime          troff
basename         fgrep            makeplot         ruusend          tsort
bdiff            file             maketemplate     rwho             unget
cal              filehassccs      maketextfile     sact             uniq
calendar         gbom             makewhatis       sccsdiff         units
canit            get              makewritefiles   sccshelp         uucp
cb               get.set          mesg             sccstorcs        uudecode
cdc              getcomp          nroff            secassign        uuencode
ci               getfile          outputstream     sepcomp          uulog
cmd1             getlink          package          seplink          uuname
cmd2             glib             pascal           shorten          uupoll
cmpassign        gphysdly         pckassign        showcomp         uuq
co               gscald           pckerrdoc        showlink         uusend
col              gspares          plottime         simassign        uusnap
com232           help             post             simulate         uux
comb             hpprintplot      prof             size             val
comm             ident            prs              sleep            verassign
compare          install          query            sort             verify
comperr          iskitok          rcs              spell            verify.test
compile          join             rcsdiff          spellin          vmcopy
copystream       led              rcsmerge         spellout         vmexec
cpq              led2stream       rev              spice            vprintplot
crypt            lex              rlog             split            what
cu232            lint             rmchg            stream2led       yacc
delta            lnkassign        rmcomp           strip
deroff           look             rmdel            sum


usr/dict:
--------

hlista  hlistb  hstop   words


usr/doc:
-------

Mail           cman           iosys          porttour       setup
README         com232.doc     lex            regen          shell
adb            ctour          lint           run            summary
adv.ed         dc             lpd            sccs           sysmgr
assembler      edtut          m4             scope          tbl
awk            eqn            make           security       uprog
beginners      implement      more.help      sed            uucp
clswinst.mem   index          password       sendmail       yacc

usr/doc/Mail:
```

```
------------

Makefile  mail1.nr  mail3.nr  mail5.nr  mail7.nr  mail9.nr
mail0.nr  mail2.nr  mail4.nr  mail6.nr  mail8.nr  maila.nr

usr/doc/adb:
-----------

tut   tut1

usr/doc/adv.ed:
--------------

ae.mac  ae0     ae1     ae2     ae3     ae4     ae5     ae6     ae7     ae9

usr/doc/beginners:
-----------------

u.mac  u0     u1     u2     u3     u4     u5

usr/doc/ctour:
-------------

cdoc0  cdoc1  cdoc2  cdoc3  cdoc4

usr/doc/edtut:
-------------

e.mac  e0     e1     e2     e3     e4     e5     e6     e7

usr/doc/lpd:
-----------

0.t  1.t  2.t  3.t  4.t  5.t  6.t  7.t

usr/doc/porttour:
----------------

porttour1  porttour2

usr/doc/sendmail:
----------------

intro.me    op.me       rfc819.lpr  rfc821.lpr  rfc822.lpr  usenix.me

usr/doc/shell:
-------------

t.mac  t1     t2     t3     t4

usr/doc/summary:
---------------

hel.mac  hel0     hel1     hel2     hel3     hel4     hel5     hel6
```

```
usr/doc/uprog:
-------------

cwscript  p0        p2        p4        p6        p9
p.mac     p1        p3        p5        p8

usr/doc/uucp:
------------

implement  network    uucpinst

usr/doc/yacc:
------------

ss..   ss1   ss3   ss5   ss7   ss9   ssB   ssb   ssd
ss0    ss2   ss4   ss6   ss8   ssA   ssa   ssc

usr/include:
-----------

a.out.h         dfs             mp.h            setjmp.h        tci
ar.bky.h        disktab.h       mtab.h          sgtty.h         utmp.h
ar.h            dumprestor.h    net             signal.h        varargs.h
arpa            efs             netdb.h         stab.h          vfont.h
as.h            errno.h         netinet         stdio.h         vnet
assert.h        fcntl.h         nlist.h         strings.h       white
bootstrap.h     fstab.h         pwd.h           struct.h        whoami.h
conn            grp.h           ranlib.h        sys
ctype.h         lastlog.h       rpc             syscall.h
curses.h        machine         s32             sysexits.h
dbm.h           math.h          s32dev          syslog.h

usr/include/arpa:
----------------

ftp.h     ind.h     inet.h    telnet.h  tftp.h

usr/include/conn:
----------------

conn.h          conn_packet.h

usr/include/dfs:
---------------

dfs.h         dfs_ioctl.h

usr/include/efs:
---------------

efs.h         efs_ioctl.h

usr/include/machine:
-------------------

busanal.h     dkio.h        mem.h         param.h       reentrant.h  setjmp.h
clock.h       dklabel.h     mtpr.h        pcb.h         reg.h        trap.h
cpu.h         frame.h       nec7201.h     psl.h         rpb.h        vectors.h
```

```
debug.h      kstat.h      necuart.h    pte.h        scCpu.h      vmparam.h

usr/include/net:
---------------

af.h      if.h      netisr.h  raw_cb.h  route.h

usr/include/netinet:
-------------------

arp.h        in.h         ip.h         tcp.h        tcp_seq.h    tcpip.h
icmp_var.h   in_pcb.h     ip_icmp.h    tcp_debug.h  tcp_timer.h  udp.h
if_ether.h   in_systm.h   ip_var.h     tcp_fsm.h    tcp_var.h    udp_var.h

usr/include/rpc:
---------------

rpc.h       rpc_stat.h

usr/include/s32:
---------------

busanal.h    dkio.h       mem.h        param.h      reentrant.h  setjmp.h
clock.h      dklabel.h    mtpr.h       pcb.h        reg.h        trap.h
cpu.h        frame.h      nec7201.h    psl.h        rpb.h        vectors.h
debug.h      kstat.h      necuart.h    pte.h        scCpu.h      vmparam.h

usr/include/s32dev:
------------------

badblock.h   exioctl.h    isvar.h      rfreg.h      smoctreg.h   vpb.h
circbuf.h    exreg.h      mbvar.h      rfvar.h      useful.h     vpreg.h
console.h    if_moreg.h   octreg.h     rim_ioctl.h  vg_ioctl.h
ecreg.h      is_ioctl.h   omni.h       screg.h      vgvar.h
exdef.h      isreg.h      regmuck.h    skyreg.h     vp.h

usr/include/sys:
---------------

acct.h       domain.h     mount.h      seg.h        trace.h      vcmd.h
bk.h         errno.h      msgbuf.h     sgtty.h      trap.h       vlimit.h
buf.h        file.h       mtio.h       signal.h     tty.h        vm.h
callout.h    fs.h         nami.h       socket.h     ttychars.h   vmmac.h
clist.h      gprof.h      param.h      socketvar.h  ttydev.h     vmmeter.h
cmap.h       inode.h      pdma.h       stat.h       types.h      vmparam.h
conf.h       ioctl.h      proc.h       systm.h      uio.h        vmsystm.h
dir.h        kernel.h     protosw.h    text.h       un.h         vtimes.h
dk.h         map.h        quota.h      time.h       unpcb.h      wait.h
dkbad.h      mbuf.h       reboot.h     timeb.h      user.h
dmap.h       mman.h       resource.h   times.h      vadvise.h

usr/include/tci:
---------------

community.h   exos_decnet.h  exos_sync.h   tci.h

usr/include/vnet:
----------------
```

```
bulkdata.h       bulkdata_stat.h  vnet.h           vnet_ioctl.h

usr/include/white:
-----------------

acia.h         diskformat.h   fd.h           kb.h           vt100cook.h
config.h       dispmgr.h      if_toreg.h     mouse.h        vt100pager.h
cvreg.h        ds.h           iobuf.h        via.h

usr/lib:
-------

Mail.help      crontab.local  getNAME        more.help      suftab
Mail.help.~    diff3          help           pascterrs      tabset
Mail.rc        diffh          hpf            rdiff          term
aliases        ex3.7preserve  lex            rdiff3         tmac
aliases.dir    ex3.7recover   lint           rscs           units
aliases.pag    ex3.7strings   lpd            sendmail       uucp
atrun          font           lpf            sendmail.cf    whatis
calendar       ftncterrs      makekey        sendmail.hf    yaccpar
crontab        ftnrterrs      merge          spell

usr/lib/font:
------------

ftB    ftCE   ftCS   ftG    ftGR   ftHM   ftLI   ftPI   ftSB   ftUD
ftBC   ftCI   ftCW   ftGI   ftH    ftI    ftPA   ftR    ftSI   ftXM
ftC    ftCK   ftFD   ftGM   ftHI   ftL    ftPB   ftS    ftSM

usr/lib/help:
------------

ad       cb       cmds     de       ge       prs      un       vc
bd       cm       co       default  he       rc       ut

usr/lib/lex:
-----------

ncform  nrform

usr/lib/lint:
------------

lint1          llib-lc        llib-port
lint2          llib-lc.ln     llib-port.ln

usr/lib/rscs:
------------

bisync         killrscs       rdaemon        rexec          trace
installrscs    notroot        restartrscs    rsleep

usr/lib/tabset:
--------------

3101       beehive    std        teleray    xerox1720
aa         diablo     stdcrt     vt100
```

```
usr/lib/term:
-------------

tab2631      tab300-12     tab300s-12    tab450       tabal
tab2631-c    tab300S       tab37         tab450-12    tablp
tab2631-e    tab300S-12    tab382        tab832       tabtn300
tab300       tab300s       tab4000A      tabX


usr/lib/tmac:
-------------

tmac.an      tmac.s       tmac.scover   tmac.sdisp   tmac.skeep   tmac.srefs


usr/lib/uucp:
-------------

L-devices    L.cmds       USERFILE      uuclean
L-dialcodes  L.sys        uucico        uuxqt


usr/man:
-------
man1   man2   man3   man4   man5   man7   man8


usr/man/man1:
-------------

Mail.1         ed.1           lpr.1          reset.1        true.1
adb.1          edit.1         lprm.1         rev.1          tsort.1
admin.1        egrep.1        ls.1           rlog.1         tty.1
ar.1           eqn.1          m4.1           rlogin.1c      unexpand.1
at.1           ex.1           mail.1         rm.1           unget.1
awk.1          expand.1       mailq.1        rmail.1        uniq.1
basename.1     expr.1         make.1         rmdel.1        units.1
bdiff.1        false.1        man.1          rmdir.1        uucp.1c
cal.1          fd.1           merge.1        rsh.1c         uudecode.1c
calendar.1     fdcopy.1       mesg.1         ruptime.1      uuencode.1c
cat.1          fdio.1         mkdir.1        rwho.1         uulog.1c
cb.1           fgrep.1        mkstr.1        script.1       uuname.1c
cc.1           file.1         more.1         sed.1          uuq.1c
cd.1           find.1         msgs.1         sh.1           uusend.1c
chgrp.1        finger.1       mt.1           shownet.1V     uux.1c
chmod.1        ftp.1c         mv.1           size.1         val.1
chsh.1         get.1          netstat.1      sleep.1        vfontinfo.1
ci.1           getline.1V     newaliases.1   sort.1         vgrind.1
clear.1        grep.1         nice.1         spell.1        vi.1
cmp.1          groups.1       nm.1           split.1        vmstat.1
co.1           head.1         nroff.1        strings.1      vpl.1V
col.1          hostid.1       od.1           strip.1        vpq.1
comb.1         hostname.1     page.1         stty.1         vpr.1
comm.1         ident.1        pagesize.1     su.1           vprm.1
cp.1           install.1      passwd.1       sum.1          vtroff.1
cpio.1         intro.1        plot.1g        tail.1         vwidth.1
cptree.1V      iostat.1       pr.1           talk.1         wait.1
crypt.1        join.1         praliases.1    tar.1          wall.1
csh.1          kill.1         printenv.1     tbl.1          wc.1
ctags.1        last.1         prmail.1       tee.1          what.1
date.1         ld.1           prof.1         telnet.1c      whatis.1
dd.1           lex.1          prs.1          test.1         whereis.1
delta.1        lint.1         ps.1           tftp.1c        which.1
deroff.1       ln.1           pwd.1          time.1         who.1
```

```
df.1           lock.1         rcp.1c         tip.1c         whoami.1
diff.1         login.1        rcs.1          touch.1        write.1
diff3.1        look.1         rcsdiff.1      tp.1           xstr.1
du.1           lorder.1       rcsintro.1     tr.1           yacc.1
echo.1         lpq.1          rcsmerge.1     troff.1        yes.1

usr/man/man2:
------------

accept.2         getgroups.2       mkdir.2          sigblock.2
access.2         gethostid.2       mknod.2          signal.2
acct.2           gethostname.2     mount.2          sigpause.2
bind.2           getitimer.2       open.2           sigsetmask.2
brk.2            getpagesize.2     pipe.2           sigstack.2
chdir.2          getpgrp.2         profil.2         sigvec.2
chmod.2          getpid.2          ptrace.2         socket.2
chown.2          getpriority.2     read.2           stat.2
chroot.2         getrlimit.2       readlink.2       symlink.2
close.2          getrusage.2       reboot.2         sync.2
connect.2        getsockopt.2      recv.2           syscall.2
creat.2          gettimeofday.2    rename.2         truncate.2
dup.2            getuid.2          rmdir.2          umask.2
execve.2         intro.2           send.2           unlink.2
exit.2           ioctl.2           setgroups.2      utimes.2
flock.2          kill.2            setpgrp.2        vfork.2
fork.2           killpg.2          setregid.2       vhangup.2
fsync.2          link.2            setreuid.2       wait.2
getdtablesize.2  listen.2          setrlimit.2      wait3.2
getgid.2         lseek.2           shutdown.2       write.2

usr/man/man3:
------------

abort.3         floor.3m        gets.3s         popen.3s        sleep.3
abs.3           fopen.3s        getservent.3n   printf.3s       stdio.3s
alarm.3c        fread.3s        getwd.3         psignal.3       string.3
atof.3          frexp.3         hypot.3m        putc.3s         stty.3c
bstring.3       fseek.3s        inet.3n         puts.3s         swab.3
byteorder.3n    gamma.3m        initgroups.3    qsort.3         system.3
crypt.3         genetent.3n     intro.3         rand.3c         termcap.3x
ctime.3         getc.3f         intro.3c        random.3        time.3c
ctype.3         getc.3s         intro.3m        regex.3         times.3c
curses.3x       getenv.3        intro.3s        rexec.3x        ttyname.3
directory.3     getfsent.3x     j0.3m           scandir.3       ungetc.3s
ecvt.3          getgrent.3      malloc.3        scanf.3s        utime.3c
end.3           gethostent.3n   mktemp.3        setbuf.3s       varargs.3
execl.3         getlogin.3      monitor.3       setjmp.3        vlimit.3c
exit.3f         getpass.3       nice.3c         setuid.3        vtimes.3c
exp.3m          getprotoent.3n  nlist.3         signal.3c
fclose.3s       getpw.3c        pause.3c        sin.3m
ferror.3s       getpwent.3      perror.3        sinh.3m

usr/man/man4:
------------

Mem.4          drum.4       is.4v       mom.4        tcp.4p      va.4v
acia.4         ec.4v        lo.4        null.4       tom.4       vp.4v
autoconf.4v    intro.4n     lp.4v       pty.4        tty.4
dkio.4v        ip.4p        mem.4       rimtape.4    udp.4p

usr/man/man5:
------------
```

```
a.out.5        dir.5         gettytab.5    phones.5      stab.5        types.5
acct.5         disktab.5     group.5       printcap.5    tar.5         utmp.5
aliases.5      dklabel.5v    hosts.5       protocols.5   termcap.5     uuencode.5
ar.5           dump.5        indaemons.5   rcsfile.5     tp.5          vfont.5
core.5         fs.5          mtab.5        remote.5      ttys.5
dbm.3x         fstab.5       passwd.5      services.5    ttytype.5

usr/man/man7:
------------

ascii.7     eqnchar.7   intro.7     man.7       term.7
environ.7   hier.7      mailaddr.7  ms.7

usr/man/man8:
------------

ac.8            dump.8          lpd.8           rc.8            sccstorcs.8
backup.8V       efsioctl.8V     makedev.8       rdump.8         shutdown.8
chown.8         ether.8V        makekey.8       reboot.8        sync.8
chuid.8V        fsck.8          makewhatis.8    reboot.8V       telnetd.8c
clri.8          ftpd.8c         mkfs.8          renice.8        tftpd.8c
cmdsw.8         getfs.8         mklost+found.8  restor.8        tunefs.8
conn.8V         getty.8         mknod.8         restore.8       update.8
crash.8         halt.8          mkproto.8       rexecd.8        uuclean.8c
cron.8          help.8          mkusr.8V        rimioctl.8V     uucp.daily.8c
cv.8            icheck.8        mount.8         rlogind.8       uucp.hourly.8c
dcheck.8        ifconfig.8c     ncheck.8        rmt.8           uupoll.8c
dk.8v           ind.8           newfs.8         rshd.8          uurate.8c
dkbn.8v         init.8          omni.8          rshsw.8         uusnap.8c
dklabel.8v      install.8       popcorn.8       rwhod.8         uuusage.8c
dktest.8v       intro.8         pstat.8         sa.8
dmesg.8         lpc.8           rbackup.8       savecore.8

usr/news/changes:
----------------

allfiles      release.date  version.name

usr/ucb:
-------

Mail        ftp         lprm        praliases   tail        wc
chsh        grep        mailq       printenv    talk        whereis
clear       groups      man         prmail      telnet      which
ctags       head        more        rcp         tftp        whoami
e           iostat      msgs        reset       tr          xstr
edit        last        netstat     rlogin      unexpand    yes
ex          lock        newaliases  rsh         vfontinfo
expand      lpq         page        script      vi
finger      lpr         pagesize    strings     view

usr/valid/bin:
-------------

com232          cptartape       getfile         rvtroff.valid   walknet
console         cptree          popcorn         shownet
cpfromvax       cu232           rvcat           tpchk
cpfromvax.sed   ftpfc           rvsort          vpl
cptape          get.set         rvtroff         vtroff.valid
```

SECTION 2
COMMUNICATIONS ENHANCEMENTS

The TCP/IP based (DARPA) networking in 4.2 is a major improvement over 4.1c and one of the main reasons for migrating to a 4.2 based system. A significant portion of the user's view of the new functionality added with 4.2 has already been implemented in the 7.27 release. The new functionality consists of the Address Resolution Protocol (ARP) that allows Ethernet controllers from various manufacturers to communicate within the same network, the FTP and TELNET programs, the UUCP program and related mail facilities, and limited support for multiple networks. The FTP and TELNET programs provide machine-independent file transfer and virtual terminal capabilities. The Berkeley *rsh*(1), *rlogin*(1), and *rcp*(1) commands can be used between UNIX machines; FTP and TELNET can be used with most vendors' machines that support TCP/IP networking. The initial 8.0 release provides the same level of internetwork capability as the 7.27 release; the 4.2 TCP/IP base allows future releases to support much more expanded internetwork capabilities. In general, the 4.2 implementations of these network features allows better compatibility with other systems, enhanced capabilities, and expandability to more varied network topologies.

There was one basic problem with TCP/IP networking under the 7.25 release of 4.1c that was corrected in by the 7.27 release. The correction of this problem has caused a compatibility problem since TCP/IP addresses for both the 7.27/4.1c and 8.0/4.2 releases now are formed by taking a network number and concatenating it with a local node number. The network number that was hardwired into the 7.25 release was network number 0 which is reserved in releases 7.27/8.0 for packets not intended for a locally known (i.e., routable) network. Accordingly, systems running 7.27 or 8.0 are unable to communicate via TCP/IP with hosts running 4.1c/7.25. Communications between a 7.25 system and a 7.27 or 8.0 system are supported via the Extended File System (EFS).

A list of the supplemental documents associated with communications enhancements follows. These documents should be inserted into the *SCALDsystem Manager's Reference Manual.*

- *A Tutorial on the TCP/IP and EFS/RPC Network Facilities* – describes TCP/IP based programs, administrative programs, and the EFS/RPC protocol.
- *Description and Tutorial on FTP* – an overview of the File Transfer Protocol.
- Description and Tutorial on Telnet – describes user interface to the ARPANET standard Telnet protocol.
- *UUCP Operation and Maintenance* – describes the setup, operation, and maintenance of the UUCP package.

- *Mail Reference Manual* – describes how to use the *Mail* program to send and receive messages.
- *SENDMAIL - An Internetwork Mail Router* – describes the *sendmail* general internetwork mail routing facility.
- *SENDMAIL Installation and Operation Guide* – describes installation and maintenance of a mail system.

The corresponding man pages for the above facilities are included in the SCALDsystem UNIX Programmer's Manual update package (900-00228 Rev A).

DECnet SUPPORT

DECnet will be supported in a future release of 8.0.

HOST FILES

Since the *chkhosts* utility updates the */etc/hosts* file on network number 0 (network 0 is reserved in release 8.0), this utility cannot be supported in the 8.0 release. The system administrator must update the */etc/hosts*, */etc/hosts.equiv*, and */.hosts* files manually.

SECTION 3
PRINTER/SPOOLER ENHANCEMENTS

3.1. NEW PRINT SPOOLER SUPPORT

The old *Vpd, vpd* and *hpd* spooler daemons have been obsoleted by the new 4.2 *lpr* print spooler (*Vpd, vpd,* and *hpd* were links to the old *lpr*; they are now shell scripts that call *lpr* appropriately). *lpr* now uses */etc/printcap* to determine printer capabilities, to handle spooling to remote machines, etc.

Please note that */etc/printcap* is shipped as */etc/printcap.proto* and must be edited to reflect the local configuration. */etc/printcap.proto* is a large file of prototype entries. The relevant entries must be uncommented, and then */etc/printcap.proto* must be moved to */etc/printcap* in order for *lpr* to function properly (see the supplemental document *Configuring a Network for Printing and Plotting*).

lpc is a new utility that controls the print spool queue by allowing users to remove and reorder entries in the queue.

A list of the supplemental documents associated with printer/spooler enhancements follows. These documents should be inserted into the *SCALDsystem Manager's Reference Manual.*

- *4.2BSD Line Printer Spooler Manual* – describes the structure and installation procedure for the line printer spooling system.

- *GED Plotting* – describes the VGB and new HPR modes for obtaining hardcopy of GED drawings.

SECTION 4
INSTALLATION CONSIDERATIONS

Due to the fact that the root file system is increasing in size, several directories have been linked into */u0* for release 8.0. */usr/bin* is now linked to */u0/usr.bin* and */usr/valid* is now linked to */u0/usr.valid.*

A number of new disk options will be introduced in the Q1 1986 time frame. Valid will be examining the standard disk partitions based on these options and the eventual expansion of virtual memory with the 68020 CPU. During Q1 1986, Valid will restructure the standard partitions, with the root partition likely growing to avoid the need to link utility directories to another file system.

4.1. SPACE REQUIREMENTS

The Valid 8.0 software release represents a change in the organization of the UNIX operating system as well as a change in the amount of space required for the operating system. The sections that follow attempt to provide some general guidelines for the amount of space required for an 8.0 system as compared to the space needed for 7.27.

In the following tables, rudimentary disk usages for various areas of software are presented. In each area, the approximate usage of 7.27 and 8.0 systems is given. In this way, you can hopefully gain a feel for the changes in space requirements necessitated by release 8.0.

4.1.1. Operating System and Utilities

For the operating system and UNIX utilities, release 8.0 requires more space than the 7.27 release; in fact, more space is required than is available on an **empty** root partition! Therefore, some of the 8.0 system binaries were moved off of the root partition. The only other partition that is uniform across Valid systems is */u0.* Accordingly, */tmp, /usr/bin,* and the new directory */usr/valid* have been moved to */u0,* and have symbolic links from root.

There must be sufficient space on */u0* for these directories as well as sufficient space on the root partition for 8.0's increased requirements.

The amount of space required on */u0* is fairly simple to calculate; */usr/bin, /usr/valid* and some space for */tmp* require the following space (in kbytes):

Space Taken by Utility Directories on */u0* in 8.0	
Kbytes	**Directory**
2100	*/u0/usr.bin* (*/usr/bin*)
150	*/u0/usr.valid* (*/usr/valid*)
2250	**total requirement for binaries on */u0***
100	*/tmp* space (an average; includes */tmp* and */u0/tmp* use)
2350	total requirement on */u0*

Your own system may use more or less space on */tmp*. Note that some space on the root partition will actually be gained with the installation of 8.0.

The following table compares a 7.27 system's disk usage with that for an 8.0 system.

7.27		8.0	
/bin	2750	*/bin*	1650
/etc	1450	*/etc*	1350
/include	850	*/include*	0
/lib	700	*/lib*	750
/usr	2350	*/usr*	3550
/vmunix	450	*/vmunix*	500
	8550		7800

On */u0,* there are certain directories linked from root that were also on */u0* in earlier releases (e.g., */u0/man* and */u0/spool* are linked to */usr/man* and */usr/spool*). The disk usage of these directories has also changed, but not too severely; their sizes are given below.

7.27		8.0	
/u0/doc	1150	*/u0/doc*	1150
/u0/fortran	500	*/u0/fortran*	550
/u0/man	1100	*/u0/man*	1350
/u0/pascal	550	*/u0/pascal*	550
/u0/spool	50	*/u0/spool*	50
/u0/src	20	*/u0/src*	20
/u0/user	50	*/u0/user*	50
/u0/vfont	4000	*/u0/vfont*	2700
	7420		6420

The difference is freed up on */u0.*

4.1.2. Applications

Like all software, the sizes of our applications tend to increase with time. Below, we examine the applications directories */u0/scald, /u0/editor,* and */u0/library* that are received by all customers. These directories contain the full complement of SCALD application software (i.e., packager, timing verifier, compiler, etc.).

7.27		8.0	
/u0/editor	700	*/u0/editor*	700
/u0/library	30	*/u0/library*	30
/u0/scald	4600	*/u0/scald*	5900
	5330		6630

/u0/layout is found only on SCALDstar systems.

7.27		8.0	
/u0/layout	3600	*/u0/layout*	2800

4.1.3. Using These Numbers as a Guide for Installation

You may use the above numbers as an approximation of the amount of space that must be available on a system being upgraded to 8.0. To determine the space usage of directories, use the **du -s** *directory_name* command; to determine the amount of free space on your disks, use the **df** (disk free) command. To determine your space requirements:

1. Figure the amount of space you are currently using for a specific application on a given partition.

2. If this number is significantly (100 Kilobytes or more) greater than the number from the table, you should not worry.

3. Otherwise, subtract your total from the table total. The difference is the amount of extra space you will need.

4. If that amount of space is currently available, then you do not need to free up any space. Otherwise, you must free up some disk space.

5. If you are **very, very** close to the amount of disk space needed, you should still free up some extra space (fragmentation may cost you some extra space you have not counted on).

4.1.4. Disclaimer

The numbers provided herein are estimates. Every effort has been made to make them as exact as possible, but every system is a little different due to fragmentation of file systems, user-modified files, and the like. If you have concerns or questions about space requirements, contact your Valid field engineer.

SYSTEM CALL INTERFACE

1. INTRODUCTION

This document describes differences in the system call interface between 4.1c and 4.2 BSD UNIX as implemented by the 8.0 release of the SCALDsystem UNIX software. Fortunately, most system calls are identical between the two; herein, we will concentrate on the differences between the two versions.

2. HOW SYSTEM CALLS WORK

Every system call entry described in chapter 2 of the *SCALDsystem UNIX Programmer's Manual* is implemented as a stub subroutine in the C library. These subroutines call the system directly by putting a *call number*, which encodes the desired system call, into register d0, and executing a trap instruction. UNIX's exception handler detects that a system call was issued, and uses the call number as an index into a table of routines. On completion of the call routine, control returns to the user's program.

Not all system calls are implemented in this way; some "calls" are really library routines that transform the arguments and then execute an actual system call. *Sbrk* is an example of such a "shadow" call. However, the basic scheme is that programs use system calls by using routines from the C library (even Pascal programs).

The system call interface defines two levels of compatibility between different versions of the system: *binary compatibility* in which the exact same programs may be run under either system; and *object code compatibility* in which the programs themselves don't need to be recompiled, but do need to be re-linked with the new C library.

3. SIGNALS

4.2 signals are radically different from those of 4.1c, and require special mention because they affect source code compatibility.

3.1. 7th EDITION SIGNALS

The classic *signal*(2) system call interface was provided by the 7th Edition UNIX and gives reliable, if buggy, service. This interface does not restart certain interrupted system calls, and resets the signal handler to the system default handler after catching a signal. This means that the catching routine has to issue another *signal* call if it wants to continue catching the signal. This signal package was supported in 4.1 in the C library as *signal*(3) and was implemented by invoking the *ossig* (old signal) system call.

3.2. 4.1c SIGNALS

In 4.1c, a new *sigsys*(2) interface, with slightly different semantics was introduced. Access to this package was via the *jobs* library. It was implemented by trapping to the *ossig* (old signal) system call.

3.3. 4.2 SIGNALS

For 4.2, *sigsys* was scrapped entirely and replaced with *sigvec*(2). *Sigvec* is considerably more flexible than previous signal packages. Unlike *signal* and *sigsys, sigvec* restarts interrupted system calls after the signal has been processed. The old *signal*(2) interface is gone; it is imperfectly emulated in the C library by *signal*(3). *Signal*(2) did not restart certain system calls while *signal*(3) restarts all system calls. More importantly, *signal*(3) **does not** reset the signal handler to the default handler after catching a signal.

4. COMPATIBILITY ISSUES

Tricky programs, certain use of signals, and selected 4.1c system calls will require you to modify your source.

4.1. TRICKY PROGRAMS

If your programs include home-grown system calls, either in your own library routines, or in calls on *syscall*(2), they will likely have to be modified.

If your program derives values from standard system header files, beware: some of the constants in *<sys/file.h>* have changed values and meanings. You will have to re-compute the constants. This especially affects Pascal programs that don't use the header files directly, but rather define the constants themselves.

4.2. SIGNALS

Any programs that use the old *signal*(2) mechanism may not be source-code compatible. If you rely on certain system calls not being restarted, you are in trouble. Further, if you rely on the fact that the signal handler will be reset after receipt of a signal, you will have to modify your program.

Any programs that use the *sigsys* interface are not source-code compatible and must be modified to use the new signal mechanism.

4.3. MISSING SYSTEM CALLS

System call #92, *wrap,* is now *fcntl* in 4.2. *Socketaddr,* system call #103, now results in a call on *nosys.* Further, neither call exists in the 4.2 C library. Fortunately, *wrap* is not implemented in the 4.1c C library, so you are unlikely to have used it.

Getdprop is not in the 4.1c C library, but the system call exists and does something. It does not exist either as a 4.2 library routine or as a system call. Thus use of *getdprop* is totally incompatible.

4.4. THE MISSING 'N'

System call #141, which results from the C library call *unmount*, is now *getpeername* in 4.2. An equivalent, but differently-named routine called *umount* exists in the 4.2 C library. No compatible routine exists in the 4.2 compatibility library. However, *unmount* is used only by the **umount** program and probably isn't a problem for you.

4.5. UNIMPLEMENTED/UNDOCUMENTED CALLS

The following 4.1c calls are in the 4.1c C library, and are legitimate system calls, but are no-ops and thus are unlikely to be called. They do not exist in the 4.2 library, but the system calls exist as no-ops. They are *mremap, sstk, mmap, munmap, mprotect, madvise, mincore, getdopt,* and *setdopt.*

Portal and *revoke* do not appear in the 4.1c C library, but the system calls exist as no-ops. They do not appear in the 4.2 library, and return "Bad system call" (SIGSYS) exceptions.

4.6. OBJECT LEVEL COMPATIBILITY

Stty(3) and *gtty*(3) exist in the 4.1 C library and exist as system calls. They exist only in the 4.2 compatibility library. Thus, if you use *stty* or *gtty* you must link with the 4.2 C library, which includes the compatibility library.

4.7. BINARY COMPATIBILITY

The C library implements system calls using the "latest" versions of the system entries. However, for binary compatibility with older systems, certain "old" calls are retained. These are *wait, time, stat, setuid, stime, alarm, fstat, pause, utime, nice, setpgrp, times, setgid, ssig, vlimit, vtimes,* and *vadvise.*

4.8. Ftime - BINARY COMPATIBLE AND IN THE COMPATIBILITY LIBRARY

Ftime exists in 4.1c, and a corresponding system call exists in 4.2. Further, the 4.2 compatibility library has an entry for *ftime,* so if you have to recompile your program for other reasons, you'll have to link with the compatibility library.

HOW CRM DIFFERS FROM VRM

1. INTRODUCTION

CRM is the new ROM monitor for Valid 68010 and 68020 processor boards. It is similar to VRM, but does have some important differences. This document highlights those differences. It is not intended as a CRM specification — for information on using CRM, see the *Combined Resident Monitor User's Manual* included with this change package.

2. CPU BOARD SWITCH PACK

The CPU board switch pack is an 8-switch DIP that is read by CRM. In VRM, switches 7, 3, and 2 were unused. These switches are now signficant to CRM.

Please note: the terminology regarding switch position is a bit confusing. In this document, when we refer to a switch being **on**, we mean that it be set so that the lever is positioned **away** from the board. This position is confusingly marked **off** on the switch pack.

2.1. SWITCH 7 - DYNAMIC BURN-IN

Setting Switch 7 on tells CRM to run a special dynamic burn-in routine normally used during the manufacturing proccess. You should never need to run the dynamic burn-in routine, so switch 7 should be off.

2.2. SWITCHES 2 AND 3 - DEFAULT DISK AND TAPE DRIVE SELECTION

CRM now supports two types of controller boards: the old Rimfire controllers, and the new Interphase controller boards. In the past, there were two types of Rimfire boards, the Rimfire 44 and the Rimfire 45. Similarly, we now support two types of Interphase boards, the 2190, which controls SMD disk drives (e.g. the Fujitsu Eagle), and the Storager, a multifunction board that controls ST506 and ESDI 5-1/4 inch disk drives and 1/4 inch cartridge tape drives.

Switch 3 is used to determine whether the default disk drive is on the Rimfire controller or on the Interphase controller. If switch 3 is off, the default drive is on the Rimfire controller. If switch 3 is on, the default drive is on the Interphase controller. If you only have one type of controller in the system, switch 3 must be set to reflect the type of controller present. If you have both types of controllers present, switch 3 should be set to the drive that you usually boot from. You can use the **SDUN** command to change the drive to be different from the one selected by the switch if need be, but it's convenient to have the default drive be the drive most frequently used.

Similarly, switch 2 is used to select the default tape drive. If switch 2 is off, the default tape drive is the one connected to the Rimfire controller. If switch 2 is on, the default tape drive is the one connected to the Interphase controller. Don't select the default tape drive to Interphase if the controller is a 2190, as the 2190 does not handle tape.

3. CRM COMMAND FORMAT

CRM still supports the old VRM command format, but features some improvements.

3.1. EXTENDED COMMAND PREFIX

Most of the extended commands do not require an **X** prefix. For example, you can type **BOOT** in the place of **XBOOT**. The extended commands that do require the **X** prefix are those that would conflict with immediate mode commands: **XLLA**, **XLR**, **XLSPE**, **XMEMT**, **XMODE**, **XPM**, **XTR**, **XTRET**, **XTRST**, **XTSTA**, and **XTW**.

3.2. ERASING AND TERMINATING COMMANDS

Backspace deletes the last character typed, and ^U (control U) deletes the entire line. Line feed, carriage return, and '.' all may be used to terminate a command. ^C (control C) and ^X (control X) may be used to terminate the current command and return to CRM command prompt.

3.3. COUNT PREFIX FOR DIAGNOSTIC COMMANDS

Except for the **XLLA** command, which loops forever, diagnostic commands can be prefixed with a count value.

4. IMMEDIATE MODE COMMANDS

There are a few new immediate mode commands. **#** restarts CRM and reintializes any devices. **V** displays the CRM version string. The memory deposit and display commands work as before. However, there is now a byte/short/long mode command, **XMODE**, that can be used to set the data quantization for deposit and display routines. This affects the **-**, **N**, **P**, and **T** commands.

5. DIAGNOSTIC COMMANDS

Almost all the diagnostic commands are new. **XLLA** and **XALLT** are carried over from VRM. All diagnostic commands except **XLLA** may be preceeded by a repetition count. *If no repetition count is supplied, the diagnostic repeats forever.* All diagnostics can be stopped with a ^C or ^X.

6. DISK AND TAPE COMMANDS

CRM now supports the new Interphase controllers, the 2190 that handles SMD disks (of which the Fujitsu Eagle is perhaps best known) and the Storager, that supports ST506 and ESDI 5-1/4 inch disks and 1/4 inch tape. Because CRM supports these new controllers, it's disk and tape commands are a bit different.

As we discussed previously, the default disk and tape drive may be selected using switches on the CPU board. The default disk and tape drive is always unit zero on the controller you select with the CPU switches. CRM looks for the presence of these devices on startup, and complains if they are not present or in the proper state (for example, if a disk does not show ready or is not formatted). Disk and tape unit selection may be changed by using the **SDUN** and **STUN** commands, respectively. If you suspect that something is wrong with the controllers or drives, you may use the **CINIT** command to re-initialize the controllers. You also need to use **CINIT** to spin up alternate drives, as the **XDUP** and **XDDN** commands are no longer present.

6.1. DISK AND TAPE UNIT SELECTION

The disk and tape commands use a new syntax to refer to a particular disk or tape drive. It is a 4-character string that is broken down as follows:

1. The first character is the controller type, which may be either **R** for Rimfire or **I** for Interphase.
2. The second character is the device type, which may be either **D** for disk or **T** for tape.
3. The third character is a digit that specifies the controller number (as most systems only have one controller of each type, this digit is almost always 0).
4. The last character is a digit that specifies the drive number on the selected controller.

At this point, a few examples are in order. In order to specify the zeroth Rimfire disk on the zeroth controller (almost always the appropriate Rimfire disk) type **RD00**. To specify the zeroth tape drive on the zeroth Interphase Storager controller type **IT00**.

6.2. NEW TAPE COMMANDS

STUN and **DTUN** set and display the default tape unit number. **TRST** and **TRET** reset the tape drive and retension the tape, respectively.

7. DUMP TAPE CREATION COMMAND

Creation of dump tapes has been a long, involved, and confusing process. CRM now employs some major advances in software engineering technology to automate making dump tapes. Merely type **COREDUMP** and a dump tape will be made on the default tape drive. You'll even be told how to boot UNIX to tar off the kernel to finish the job!

8. DISCONTINUED COMMANDS

8.1. DISCONTINUED IMMEDIATE MODE COMMANDS

The =, **Z**, (,), and * immediate mode commands are no longer supported.

8.2. DISCONTINUED EXTENDED COMMANDS

The bus analyzer board is no longer supported in the ROM monitor. Thus the **X?BAB**, **XIBAO**, **XLBAO**, **XLBP**, **XDBP**, **XBBA**, and **XEBA** commands are not supported.

Down-loading from the CPU host port is no longer supported, and the **XBDL** and **XEDL** commands have gone away.

Disk extended commands have been generalized and reorganized. The **X?RF** command has become **X?DK**. The **XDNOP**, **XDPB**, **XDREC**, **XDUP**, **XDDn**, **XMAPd**, **XSEEK**, **XRID**, and **XDCMD** commands are not supported. Disk formatting is supported by a stand-alone utility.

Finally, the tape commands **XTNOP** and **XTPB** are not present.

COMBINED RESIDENT MONITOR USER'S MANUAL

1. INTRODUCTION

CRM is the new ROM monitor for Valid 68010 and 68020 CPU boards. This document contains a description of the commands available on CRM. To execute CRM commands, the operator must use a terminal connected to the console port on the CPU.

CRM continuously listens to the console port for commands and executes whatever it sees and understands. CRM echos everything it receives from the console port except most control characters. The host port is currently not used by CRM.

Upper-case and lower-case characters are equivalent in CRM commands. CRM has a rotating 4-byte accumulator. A valid hex digit pushed into the accumulator on the right causes a hex digit to fall out on the left. CRM also has an address register that is loaded from the accumulator by command. Various commands use the address register as a parameter. CRM normally intercepts traps and prints the trap number and other critical information.

In this document '**0x**' preceding a number signifies a hexadecimal value, a ''byte'' is an 8-bit quantity, a ''word'' or a ''short'' is a 16-bit quantity, and a ''long word'' or a ''long'' is a 32-bit quantity.

2. SWITCH SETTINGS

The eight switches on the CPU board are read by CRM on reset. A switch is considered **on** when it is positioned away from the board.

Switch Settings (on, off, ? = user's choice, sys = system dependent)	
Number/Setting	**Meaning When Switch Is On**
8 (?)	Autoboot UNIX
7 (off)	Enter Dynamic burnin-mode on power up/reset
6 (?)	Run extended diagnostics on power up/reset
5 (on)	CPU has 16Meg map (should always be on)
4 (on)	Enable low Multibus memory access (off = Disable access)
3 (sys)	Default DISK drive on Interphase controller (off = Rimfire controller)
2 (sys)	Default TAPE drive on Interphase controller (off = Rimfire controller)
1 (off)	UNUSED

Note: CRM takes the CPU board out of boot state at reset time only if low Multibus memory exists and switch 4 is set accordingly.

3. CRM COMMANDS

The CRM command parser accepts the following syntaxes:

```
immediate_command
[count]immediate_command
non-immediate_command
[count]non-immediate_command
```

All commands may start with *X*. The **X** is optional unless it is necessary to resolve a conflict with an IMMEDIATE command. For example, "**ALLT**" would be interpreted as the immediate command sequence **AL L T** (load the accumulator with 0xA, load it again with 0xA then display address 0xA) rather than **XALLT** (execute "all" diagnostics) command.

Commands that could be interpreted as a count preceding a non-immediate command are resolved in favor of the non-count command (e.g., **BOOT** is correctly interpreted as **BOOT** instead of (0xB)**OOT**).

Backspace deletes the last character typed, and ^U deletes the entire line. Linefeed, Return, and '.' may all be used to terminate a command. ^C and ^X have the same function, namely to break out of the current command and return to the menu.

If CRM does not understand the extended-command name, it does nothing and suggests that you "**Type ? for help !**."

3.1. BAUD RATE SELECTION

CRM assumes a baud rate of 9600 baud at power up. The baud rate may be changed to match most terminals by hitting the BREAK key repeatedly until the correct baud rate is achieved (the Esprit II terminal requires SHIFT-BREAK). Each time the BREAK key is hit, the next rate is selected from the table below:

Baud Rate Selection
9600
19200
50
75
110
134.5
150
300
600
1200
1800
2000
2400
3600
4800
7200

3.2. SINGLE-KEYSTROKE (IMMEDIATE) COMMANDS

#	Restart CRM, reinitializing any devices and ECMs.
L	Load the address register. The long word in the rotating accumulator is loaded into the address register, then the rotating accumulator is set to zero.
-	Store a byte, short or long depending on current mode. The byte/short/long in the rotating accumulator is stored at the logical address currently in the address register.
N	Display contents of next address. The address register is incremented, and then the byte/short/long pointed to by the address register is read and displayed.
P	Display contents of previous address. The address register is decremented, and then the byte/short/long pointed to by the address register is read and displayed.
T	Display contents of this address. The byte/short/long pointed to by the address register is read and displayed.
J	Jump to the address pointed to by the address register.

M	Display a 64-byte block. A 64-byte block of memory is displayed starting at the next lowest 64-byte boundary from the address in the address register.
V	Display CRM version (number and date). Also show the default disk and tape drives and the most recent exception along with corresponding fault PC and SP.

3.3. BASIC EXTENDED (NONIMMEDIATE) COMMANDS

?	Print menu of basic extended commands.
!	Echo input to console terminal.
COREDUMP	Make a 16M crash tape on the default tape drive.
XBOOT	Boot UNIX. You will be asked for a boot device; **P** - Primary stands for the primary boot block on your disk, and is the one used most often. The other options are for booting when the primary area is corrupt. The next prompt asks for the startup mode; **A** - Auto means it will boot all the way to multi-user (asking a few questions along the way), **P** - Parms means it will ask you which UNIX to boot and bring that UNIX up single user. Generally you'll want to boot: **rd(0,6)vmunix**. From single-user UNIX type **^D** to start multi-user UNIX.
XLSPE	Load scratch-page entry.
XMODE	Set mode (i.e., the number of bytes per R/W access) to one of byte/short/long. Commands affected are the immediate commands '**N**,' '**P**,' '**T**,' and '**-**' and the non-immediate command '**XLLA**.' Note that **XMEMT** does not use mode to determine unit of access; it always performs longword accesses.
?TST	Print diagnostic commands.
?ECM	Print ECM commands.
?GEN	Print generic disk/tape commands.
?DK	Print disk commands.
?TP	Print tape commands.

3.4. DIAGNOSTIC COMMANDS

Whenever a diagnostic is executed, CRM attempts to light the red LED and to print an 'F' if it fails, and attempts to light the green LED and to print a 'P' if it passes. ALL diagnostic commands (except **XLLA**) can be preceded by a repetition count. If no repetition count is specified, the command is executed indefinitely. All diagnostics (except **XLR**) can be prematurely terminated by typing ^C or ^X.

XLLA Loop on logical address (write or read). **XLLA** reads or writes byte/short/long data depending on the current MODE (set by **XMODE**). **XLLA** prompts for:

```
R/W ?
address ?
data ? (only if Write)
silent ?
```

If silent is specified, **XLLA** loops forever on the address (the only way to abort is by doing a hardware reset). Before each read or write access, the red LED on the CPU is turned on and off once (for scope triggering purposes). If silent is not specified and if R (read), the address and data read are displayed after each access. If W (write), the address and data are displayed once. To abort, type ^C or ^X.

XSR Status register test. All bits in the status register are checked. The test is non-destructive.

XCR Context register test. All bits in the context register are checked. The test is non-destructive.

XSM Segment map test. All locations in the segment map are tested using a semi-random pattern. A big map (16k) is assumed. After testing, the segment map is reinitialized to the identity map.

XPM Page map test. All locations in the page map are tested using a semi-random pattern. A big map (16k) is assumed. After testing, the page map is reinitialized to the identity map.

XCTC Counter timer chip test. An internal location in the timer chip is tested.

XUARTS Console and host UART test. For each UART, an internal location is tested.

XSW Switch test. The switches are not allowed to be all zeroes or all ones. This is checked.

XLR Local RAM test. Execute local RAM diagnostic (destroying save area).

XALLT Execute "all" diagnostics. The following diagnostics are executed: (**SR**, **CR**, **SM**, **PM**, **SW**, **CTC**, **UARTS**) The sequence of tests is repeated a number of times taken from the rotating accumulator. The **XALLT** command is automatically executed at reset time only if the diagnostics switch is turned on.

3.5. ECM COMMANDS

XDES Display status of every Error Correcting Memory (ECM) board.

XIECM Initialize every ECM board.

XECMT Test every ECM board using signature analysis. The XECMT command is automatically executed at reset time only if the diagnostic switch is turned on.

XMEMT Execute memory-test diagnostic. Prompts user for start and end address (does longword accesses).

3.6. GENERIC DISK/TAPE COMMANDS

XCINIT Initialize the controllers.

XBKUP Backup the current disk. CRM asks for the start cylinder (default = 0), start head(0), start sector(1) and end cylinder (default = 0). To backup a 70-Mbyte drive on 1/4-inch tape, put cylinders 0x0 through 0x202 on one tape and repeat **XBKUP** with cylinders 0x203 through 0x3F0. For 1/2-inch tape, use 0x0 to 0x3F0. A 2400-foot reel of tape holds about 40 Mbytes. If you have more than one disk and you want to "XBKUP" a disk other than disk 0, you must change the default disk unit number with the **XSDUN** (set disk unit number) command. Note also that there is a **XDDUN** (display disk unit number) command.

XRSTR Restore the current disk from tape. This command asks for the start and end cylinders like the **XBKUP** command above; the same cylinder numbers used in **XBKUP** should be entered.

3.7. GENERIC DISK COMMANDS

XSDUN Set disk unit number. CRM asks for the number you want as the new default disk unit number (CRM initially defaults to RD00). The disk unit number is broken into the following four fields:

1. Controller type (R = Rimfire, I = Interphase)
2. Device type (D = Disk)
3. Controller number (system dependent)
4. Device number (system dependent)

XDDUN Display the current unit number.

XDSTA Disk status.

XDR Disk read command.

XDW	Disk write command.
XDRST	Disk reset command.

3.8. GENERIC TAPE COMMANDS

XSTUN Set tape unit number. CRM asks for the number you want as the new default tape unit number; CRM initially defaults to RT00 (or IT00 if switch 2 is on). The tape unit number is broken into the following four fields:

1. Controller type (R = Rimfire, I = Interphase)
2. Device type (T = Tape)
3. Controller number (system dependent)
4. Device number (system dependent)

XDTUN Display tape unit number.

XTSTA Tape status command.

XTR Tape read command. CRM first asks for a "start bus address" (enter a memory address in hex) and then asks for a "limit bus address" (enter another memory address in hex). The "default buffer size" message is then displayed; entering 0 defaults to 4K bytes, entering any other value sets the buffer size to that number (hex).

XTW Tape write command. CRM asks for a "start bus address" (enter a memory address in hex) and then asks a "limit bus address" (enter another memory address in hex). The "default buffer size" message is then displayed; entering 0 defaults to 4K bytes, entering any other value sets the buffer size to that number (hex).

XTRST Reset the tape drive.

XREW Rewind the tape.

XTRET Retension the tape.

XWFM Write a file mark.

4. ALPHABETICAL LISTING OF CRM COMMANDS

Command	Description
!	echo input to console terminal
#	restart CRM
-	store a byte, short or long depending on current mode
?	print menu of basic extended commands
?DK	print disk commands
?ECM	print ECM commands
?GEN	print generic disk/tape commands
?TP	print tape commands
?TST	print diagnostic commands
J	jump to the address pointed to by the address register
L	load the address register
M	display a 64-byte block
N	display contents of next address
P	display contents of previous address
T	display contents of this address
V	display CRM version
XALLT	execute "all" diagnostics
XBKUP	backup the current disk
XBOOT	boot UNIX
XCINIT	initialize the controllers
XCOREDUMP	create a 16M crash tape on default tape drive
XCR	context register test
XCTC	counter timer chip test
XDDUN	display the current disk unit number
XDES	display status of every ECM board
XDR	disk read command
XDRST	disk reset command
XDSTA	disk status
XDTUN	display tape unit number
XDW	disk write command
XECMT	test every ECM board using signature analysis
XIECM	initialize every ECM board
XLLA	loop on logical address (write or read)
XLR	execute local RAM diagnostic (destroying save area)
XLSPE	load scratch-page entry
XMEMT	execute memory-test diagnostic
XMODE	set mode to one of byte/short/long
XPM	page map test
XREW	rewind the tape
XRSTR	restore the current disk

XSDUN	set disk unit number
XSM	segment map test
XSR	status register test
XSTUN	display tape unit number
XSW	switch test
XTR	tape read command
XTRET	retension the tape
XTRST	reset the tape drive
XTSTA	tape status command
XTW	tape write command
XUARTS	console and host uart test
XWFM	write a file mark

SVS PASCAL AND SVS FORTRAN VERSION 2.4 RELEASE NOTES

The following changes have been implemented in the 2.4 release of the optional SVS FORTRAN Compiler and Symbolic Debugger supported by the 8.0 release of SCALDsystem UNIX.

1. CALLING CONVENTION CHANGES

1. **SVS FORTRAN, Parameters of Character Data Types**
   SVS FORTRAN now passes character parameters as a 4-byte pointer followed by a 4-byte length field. This is a change from previous versions of SVS FORTRAN that passed character data type parameters as a 4-byte pointer followed by a 2-byte length. The change has been made to facilitate calling 'C' subroutines from FORTRAN, since 'C' has no parameter data type corresponding to a two-value parameter. NOTE: all FORTRAN object code compiled using earlier versions of the compiler must be recompiled if it is to be linked with object code compiled using the Version 2.4 compiler. NOTE: parameter types of routines called by FORTRAN must be adjusted (check and correct all Pascal and assembly language called by FORTRAN).

2. **SVS Symbolic Debugger, FORTRAN Walkback Implemented**
   The walkback feature of the Debugger is now implemented for both FORTRAN and Pascal. Under earlier versions, the Debugger was unable to walkback through FORTRAN subroutines. The walkback feature used to determine from where the currently-active breakpoint was called (and where that point was called from, etc.). Similarly, the "up" and "down" features of the Debugger that allow access to varibles of the calling environments are now implemented in both FORTRAN and Pascal.

3. **SVS Symbolic Debugger, Printing All Fields of Records**
   The command:

   p recname.*

   prints all of the fields of the Pascal record recname. In the left column, the Debugger indicated field "offsets" (including bit position if applicable) so that it is readily apparent what values are "overlaid" for unions or variant records. The name of the record may be a complex variable such as in the command:

   p roots[23]^.llink.^.rlink.*.

4. **SVS Symbolic Debugger, Default for Setting and Clearing Breaks at Lines**
   The "break at line" and "clear break at line" commands now default to the procedure that is the current environment for the Debugger. Thus, when debugging at a breakpoint in a procedure named very_long_procedure_name, it is no longer necessary to type the command:

```
        b 1 5 very_long_procedure_name
```

since the shorthand command:

```
        b 1 5
```

is accepted by the Debugger. The current environment for the Debugger may be moved around using the "up" and "down" commands. Thus, it is convenient to set breakpoints in the current context, as well as the contexts from which the current breakpoint is called.

5. **SVS FORTRAN, Charequ Option Available on Command Line**
   The FORTRAN compiler now accepts +charequ (lower case) as a command line option in addition to the column 1 $charequ compiler directive. The effect is the same, see the language reference manual for details.

6. **SVS FORTRAN, Bitwide Logical Operations on Integers**
   SVS FORTRAN now contains four new intrinsic functions: IOR, IAND, INOT, and IEOR, that perform bitwise logical opertations in integers of length 1, 2, or 4. The opertations are inclusive or, logical and, bit complement, and exclusive or. Code is generated in line for these operations for maximum efficiency. The intrinsic functions coerce the shorter operand to the length of the longer operand by sign extension. The resultant type of the intrinsic is an integer of length equal to the longest operand. The following example illustrates the syntax and length coercion rules:

```
      program bitops

C     Need integer variables of a variety of lengths, integer*4
C     is the same as integer without length attribute since $INT2 C
C     not set.
      Integer*4 j4,k4
      integer*2 i2,j2
      integer*1 i1,j1,k1
C     Set some initial values using hexadecimal constants.
      i1 = $55
      j1 = $0f
C     Illustrate each intrinsic. Could be length 1,2, or 4.
C     Set k1 to $5f by performing inclusive or on a single byte.
      k1 = ior(i1,j1)
C     Mixed length examples.
      i2 = $5555
      j2 = $0f0f
      j4 = $0f0f0f0f
C     Set k4 to $00000f5f. This results from sign extending i1 to
C     match j2 in length, performing the two byte inclusive or and
C     then sign extending the result to four bytes for assignment
C     into k4.
      k4 = ior(i1,j2)
C     Set k1 to $05. The and is performed on four bytes after sign
C     extension of i1. The least significant byte is stored in k1.
      k1 - iand(i1,j4)
C     Set k4 to $ff ff aaa8, notice that the intrinsics may be nested
```

```
C       and that the resultant type is integer. The $0f byte is
C       complemented to $f0, sign extended to $ff f0, exclusive or'ed
C       with $5555 resulting in $aaa5 to which 3 is added. The
C       assignment to k4 is done by sign extension, in conformance
C       with the normal rules for arithmetic assignment.
         k4 - ieor(i2,inot(jl)) + 3
        end
```

7. **SVS Fortran, ISHIFT Operation on Integers**
   The ISHFT intrinsic has been added to FORTRAN. It shifts its first argument the number of places indicated by the second argument (greater than zero, left shift; less than zero, right shift). Vacated bits are filled with zeros. The following program is an example of the ISHFT intrinsic:

```
        program foo
        i = 256
        do 200 k =-10,10
        j = ishft(i,k)
        print*, 'ISHFT(',i,',',k,') is ',j
200 continue
        end
```

8. **SVS FORTRAN, Data Initialization for Named Common**
   Data statements may now initialize variables in named common blocks anywhere as an extension to the ANSI Standard which restricts this class of initialization to Block Data subprograms. If the +a "ansi only" compiler directive is specified, the ANSI Standard restriction is enforced.

9. **SVS FORTRAN, Commons of Different Sizes Allowed**
   The ANSI Standard for FORTRAN requires each specification for a named common block to be the same number of bytes. Earlier versions of SVS FORTRAN issued an error on detection of violations of this rule. Beginning with version 2.4, SVS FORTRAN only issues a warning on detection of this nonconformity with the ANSI Standard (unless the +a "ansi only" command line option is set, in which case the violation continues to be an error). The largest common size encountered becomes the size of the resulting data area.

10. **SVS FORTRAN, Argument Type Mismatching Optionally Allowed**
    The ANSI Standard requires actual and formal parameters to match in type and number. Under some, but not all circumstances, SVS FORTRAN is able to detect violations of this rule. Normally this generates an error, but the error can be altered to a warning in the case of mismatched types by specifying the $noargcheck compiler directive. The error action can be restored by specifying the $argcheck directive. In any case, if the +a "ansi only" command line option is set, type mismatches generate an error condition.

11. **SVS FORTRAN, Debugging Source Code Option**
    Except when the +a (ansi only) command line option is set, FORTRAN now accepts, in addition to the * and upper or lower case C, an upper or lower case letter D in column 1 as a comment indicator. When compiling with the -dc option ("D's note comments") the column 1 D is replaced by a blank. Thus, debugging code can be toggled on and off using a compile time command line option.

2. EXTERNAL REPRESENTATION CHANGE

1. **Character Length of Fortran Formated Records**
   Direct access, formated records in FORTRAN no longer count the "end of line" character as part of their length. That is, under previous versions, a direct access, formated file opened with a record length of 120 would only accommodate 119 characters "within" each record (118 under CPM/68k). From now on, such files allows the entire record length to be used by data characters; the system will add on, without being counted, either one or two end-of-line characters as appropriate. NOTE: the recl option in FORTRAN open and/or inquire statements may need to be adjusted in application programs to conform to the new conventions. The change has been made to eliminate arguments with the GSA over certification of SVS FORTRAN, although it is absolutely clear from the ANSI Standard that either convention is 100% standard conforming, including agreement by the GSA each time their contrary ruling has been appealed.

3. EFFICIENCY IMPROVEMENTS

1. **Code Quality**
   In a continuing effort to improve the languages, another round of optimization has been implemented. This will be particularly evident in FORTRAN, although the improvements are included in all of the language systems. An improved register management scheme has been implemented for the 68000, as well as a variety of other optimizations.

2. **Transcendental Functions and other Numeric Speed Improvements**
   The speed of the transcendental functions, as well as a few other operations in the floating point library, have been significantly improved without affecting their numeric accuracy.

3. **SVS FORTRAN, Improved Entry Code Efficiency**
   Subroutines and functions now make more extensive use of the stack for storage of local variables and parameters. The "calling conventions" are unchanged from the user's viewpoint, but more efficient calls result. Note: under Version 2.3 and earlier versions, most FORTRAN local varibales (including arrays) were static regardless of whether the program contained SAVE statements. Under Version 2.4 and subsequent versions, most local variables permitted to by dynamic by the ANSI Standard disappear on exit from procedures.

A TUTORIAL ON THE TCP/IP AND EFS/RPC NETWORK FACILITIES

1. INTRODUCTION

This document provides a tutorial explanation of the networking facilities supported by Valid. Specifically, this document discusses the Berkeley TCP/IP based programs, the new administrative programs, and the VALID EFS/RPC protocol for the 7.27 and 8.0 releases of SCALDsystem UNIX.

VALID supports network communication across its entire product line via bus-oriented multiple access channels (i.e., Ethernet and Omninet). This document is a tutorial on how to use and administrate the VALID network and its capabilities and limitations. Examples are given that use sample command lines. Command lines are shown indented from the normal text. Characters that are typed by the system are in normal type, and characters entered by the user are in **boldface** type. The following is an example of a "who" command from the system, "sartre":

```
    sartre % who
    sas    maint    May 20 10:42
```

Note the the '%' prompt is used to denote normal (non-root) commands and the '#' prompt is used to denote root commands.

2. NETWORK INTERFACES

Much use is made of the phrase ***network interface*** throughout this document. A network interface is the hardware and software that form the boundary between a network and a computer (in our case a SCALDsystem). The software that runs on the system is generally called the *driver*. The software driver and hardware form a matched set. The driver itself has a name by which it is known to the rest of the system. Most of the time, the driver name is used with the hardware as well. For example, the driver for the 3Com Ethernet controller (3C400) is known as the *ec*-driver (where *ec* is the name of the driver). The names 3Com board and *ec*-board are generally assumed to refer to the same piece of hardware: the 3C400. Currently, VALID supports three network interfaces:

ec	3Com Ethernet Controller (3C400)
mo	Multibus Omninet Transporter
to	SCALDsystem IV Omninet Transporter

With the exception of the SCALDsystem IV, all SCALDsystems can support multiple network interfaces (more than one network connection to possibly more than one network).

3. PROTOCOLS

The Valid UNIX operating system keeps track of the various network interfaces through software state. The networking implementation developed at Berkeley is layered so that the protocol layer above the network interface layer sees only a "pointer" to the network interface. In a similar manner, the network interface sees only a "pointer" to the protocol layer. The various protocols choose which network interface "pointer" to use based on their method of addressing or routing. The scenario can be described as follows:

> A protocol wants to send data in the form of a packet to another system on the network. It decides, based on the pointers provided by the network interfaces, which interface is the correct one on which to transmit and then hands the packet to the interface for transmission. The interface driver then encapsulates the packet in a network specific manner (e.g., using the Ethernet protocol) and transmits. When an interface receives a packet, it checks a type field in the packet to determine which protocol the packet is destined for. The packet is then delivered to the next highest layer for further processing and possible delivery to a user process.

VALID supports two different "transport" protocols: the Internet Protocol (IP) and the Remote Procedure Call (RPC) protocol. IP provides support for the Berkeley networking utilities (e.g., *rlogin(1)* and *telnet(1)*), while RPC supports the VALID Extended File System that provides transparent file access to files on other VALID systems.

4. ADDRESSING

Each network interface also has exactly one ***hardware address*** that is used by the hardware (Ethernet or Omninet) protocol. Each interface contains an IP *protocol address.* This is the method that IP uses to determine the network interface to be used on transmit.

IP and RPC use different methods for addressing. The method used by IP is well-known and is defined in appendix A. In short, each system (or *node*) has a 32-bit address of two parts: network and local. The network address can be 8-, 16-, or 24-bits long depending on the value of the first two bits. The local part of the address is the remaining 24-, 16-, or 8-bits, respectively. IP addresses must be unique (i.e., no two hosts attached to the same set of connected networks may have the same IP address). RPC uses an 80-bit address of which only 48-bits are used. In general, the RPC address corresponds to the hardware address of the interface (padded out if necessary).

Example:

	Interface	IP Address
mo0	(*mo* interface #0)	200.0.1.4
ec0	(*ec* interface #0)	42.0.0.11

If the IP layer is trying to send a packet to IP address 200.0.1.37, it checks each of the known interfaces to see if its address is on network 200.0.1. In our case, the *mo0* interface has an address that is on network 200.0.1. The IP layer then hands the packet to the *mo* interface driver for transmission.

5. TOPOLOGY

Each protocol has a *topology* or physical domain (connected networks, point-to-point links, etc.) that it supports. IP is supported over multiple Ethernet/Omninet networks. Any node that has more than one network interface can forward IP packets between the networks to which it is connected. Using the example above, if the *ec0* interface receives a packet destined for address 200.0.1.27, the packet is forwarded by retransmitting it through the *mo0* interface. RPC is supported only in a single-network topology. If a node has more than one network interface, it must decide which interface to use for RPC communication.

6. ROUTING

Routing is based on a very simple idea: if a system cannot get a packet directly to its destination (i.e., the destination is not directly connected to the same network cable as the local system), it must give it to some other system that can. One might ask how a system can determine what other systems are qualified to send a packet to a destination not directly reachable. Each system maintains its own routing table. Each entry in the table contains three fields: a destination, a gateway (the node that can forward the packets to the destination), and the interface that should be used. The routing table is maintained by the utility *routed*(8) described later in this document. An interesting idea is that of the wild-card router. This is a gateway to which packets are sent if no other route can be found. The wild-card router is discussed later in this document. It should be noted that RPC does not forward packets across networks.

7. NAMES AND ADDRESSES

Most people prefer using names to addresses when referencing a machine. Example:

```
sartre % rlogin darth
```

is much easier to remember than

```
sartre % rlogin 42.0.0.40
```

Each machine (SCALDsystem I, II, star, etc.) is assigned a name through the */etc/configuration* file. The name is set each time the */etc/mkconfig* program is run by the Valid Field Engineer. How, therefore, do we get protocol addresses from names? There are two answers to this question; one for each of the transport protocols (RPC and IP) that are supported. RPC uses a "connection table" that holds name-to-address bindings. Each systems broadcasts a packet on its "RPC cable" that states what name and address it is using. These packets are commonly referred to as connection packets or "conn packets." When an RPC request is made to a named node, RPC looks up the name in the table to find the corresponding address.

The IP case is more complicated because of the more complex topology. IP name-to-address bindings are stored in static files. The most familiar of these files is */etc/hosts* or the "hosts file." This file contains entries corresponding to each of the known *network interfaces* (not hosts). Each entry consists of four fields: an IP address, a name, an alias list, and an optional comment field. The IP address is assigned by a central administrator in a method that will be described further on. The name depends on the configuration of the host. If the host as exactly one network interface, the name should be the name of the host. If the host has more than one network interface, the address corresponding to the "main" interface should have the same name as the host and the other addresses should have the name of the host followed by a hyphen, followed by the name of the corresponding network interface.

An example is in order here. The following is a description of a small network:

```
Host "sartre" has one interface:
      ec0 which has IP address 42.0.0.10
Host "sophie" has two interfaces:
      mo0 which has IP address 200.0.1.4
      ec0 which has IP address 42.0.0.11
Host "poptart" has one interface:
      to0 which has IP address 200.0.1.17
Host "papa" has two interfaces:
      mo0 which has IP address 200.0.4.3
      ec0 which has IP address 42.0.0.94
```

The main interface on "papa" is *mo0*; the main interface on "sophie" is *ec0*. The following hosts file describes the network:

```
42.0.0.10       sartre
200.0.1.4       sophie-mo0
42.0.0.11       sophie-ec0 sophie
200.0.1.17      poptart
200.0.4.3       papa-mo0 papa
42.0.0.94       papa-ec0
```

The Berkeley IP utilities use library routines that try to match a name given by the user (as in the rlogin example above) to one of the addresses in the hosts file. If a match cannot be made, an error results. Some of the utilities (i.e., *telnet* and *ftp*) allow the user to give an address as well as a name. In this case, the name-address map need not be present in the hosts file. Administrative programs that get data from the kernel also try to match up addresses to names. If an address cannot be found for a name, the address is printed. Examples are given later to further clarify this point.

In addition to host name-address mappings, the Berkeley utilities support network name-address mappings. These maps are kept in the */etc/networks* or "networks file." The format of the networks file is similar to that of the hosts file. Each entry has four fields: a network name, an IP address, an alias list, and a comment field. For example

```
network "valid-engr" has IP address 42.xx.xx.xx
network "sasnet" has IP address 200.0.1.xx
network "catonet" has IP address 200.0.3.xx
```

would produce the following networks file:

```
valid-engr 42   valid_engineering     # Ethernet
sasnet          200.0.1   scotts_net       # Omninet
catonet         200.0.3   kathys_net       # Omninet
```

Two other name-address maps are kept: */etc/services* and */etc/protocols*. These files are used by the Berkeley programs to find protocol and port numbers for various well-known processes (e.g., the rlogin daemon). These files are relatively static and may be ignored by the user.

The next question we are faced with is the following: How do we make sure that all the copies of the hosts and networks files on all the machines in the connected set of networks are consistent (i.e., names and IP addresses are "correct")? The */etc/chkhosts* program that provided dynamic updates to the hosts file on a limited (single-cable) basis has been deleted from release 8.0 (chkhosts used the reserved network address 0). Until there is a better solution to the multiple network problem, hosts and networks files must be individually maintained on each machine by the system administrator (a standard version of the hosts and networks files should be kept by the network administrator for distribution). Each system is responsible for staying in sync with the standard version.

In summary, name-address mappings are kept in static files on each machine. These files are used by the Berkeley TCP/IP utilities. Each system is responsible for maintaining its own (correct) copy of these files. The standard copy is kept by the network administrator.

8. ADDRESS RESOLUTION PROTOCOL

An area that is not normally seen by the user is the conversion of protocol addresses into hardware addresses. The Address Resolution Protocol, or "ARP" as it is called, maintains a table that maps IP protocol addresses into hardware addresses to allow IP protocol addresses to be independent of the hardware address. In the past, the IP address was defined to be network 0 (8-bits) and the local address was defined to be the lowest 24-bits of the Ethernet (or Omninet padded to 24-bits) address. ARP allows Valid systems to communicate via IP with other, non-VALID systems. ARP is not used by the Valid RPC protocol.

9. STARTUP

Each system must go through initialization before it can use the networking facilities. Part of this initialization is done by the kernel at boot-time. The kernel initialization involves resetting and configuring the board(s), attaching the driver(s) to various tables, and attaching the interface data structures to queues. The rest of the initialization is done in user space, generally through the *rc* and *rc.network* files. *Rc* is the multi-user startup file. *Rc.network* is called from *rc* and performs network-specific initializations.

10. THE IFCONFIGURE PROGRAM

The *ifconfig*(8) program configures the interface structures. It can be used to set (or reset) the address of an interface, to set (or unset) structure flags, and to configure the interface used by the RPC protocol. *Ifconfig* is invoked as follows:

ifconfig *ifnet* [***address***] [***parameters***]

where *ifnet* is the name of the interface structure and *address* corresponds to the rules described above ("Names and Addresses"). Using "sartre" as an example, the following command sets the address of the interface structure:

```
sartre #  ifconfig ec0 sartre
ec0: 42.0.0.10 flags=3<UP,BROADCAST>
```

assuming that "sartre" is in the hosts file. The line typed by the system shows how the interface is configured and the value of the structure flags. Using the ***hostname*** command, a general statement can be defined that can be used on any machine that has exactly one *ec0* interface and no other interfaces:

```
sartre #  ifconfig ec0 'hostname'
ec0: 42.0.0.10 flags=3<UP,BROADCAST>
```

For "sophie" (*mo0* address 200.0.1.4, *ec0* address 42.0.0.11), the following commands configure the IP interfaces:

```
sophie # ifconfig ec0 'hostname'-ec0
ec0: 42.0.0.10 flags=3<UP,BROADCAST>
sophie # ifconfig mo0 'hostname'-mo0
mo0: 200.0.1.4 flags=43<UP,BROADCAST,RUNNING>
```

These commands do not address the problem of selecting the network interface for RPC. A special option has been added to *ifconfig* to allow selection of the RPC interface: "vnet". The following command sets both the IP address and the RPC network interface (to be *ec0*):

```
sartre # ifconfig ec0 'hostname' vnet
```

For systems with multiple interfaces (e.g, "sophie"), the *rc.network* file is built from an installation script. The following is an excerpt from sophie's *rc.network* that was built from an installation script. The network interface to be used for RPC is the *mo0* (Omninet) interface:

```
: 'Configure the interfaces'

ifconfig ec0 'hostname' vnet
ifconfig mo0 'hostname'-mo0
```

In summary, *ifconfig* sets both the interface address and flags as well as the RPC network configuration. It is generally only used in *rc.network* or single user by privileged (root) users.

11. THE NETSTAT COMMAND

The *netstat*(8) command is an important tool in monitoring the state of a system's network. In particular, *netstat* is useful for looking at the IP sub-system. *Netstat* reads through the kernel data structures to find the information requested by the user. This section discusses the various options of *netstat* and how the information gathered can be interpreted.

The **-a** option shows the state of all *sockets* that are currently allocated. Sockets are the end-points of communication between Valid UNIX processes. For example, when a user types:

```
sartre % rlogin darth
```

a process (*rlogin*) is invoked to try to rendezvous with the rlogin daemon (*rlogind*) on darth. This daemon is at a *well-known* socket address. The socket address is a triple: protocol address, sub-protocol number, and port number. An example of a sub-protocol is Transmission Control Protocol (TCP). TCP is a connection-oriented protocol built on top of IP. The **-a** option shows all currently allocated sockets and, in the case of TCP sockets, the TCP state that they are currently in. The following is the result of a "netstat -a" on sartre:

```
Active connections (including servers)
Proto   Recv-Q  Send-Q  Local Address   Foreign Address   (state)
tcp     0       0       sartre.1025     sophie.telnet     ESTABLISHED
tcp     0       0       *.telnet        *.*               LISTEN
tcp     0       0       *.ftp           *.*               LISTEN
tcp     0       0       *.login         *.*               LISTEN
tcp     0       0       *.shell         *.*               LISTEN
udp     0       0       *.521           *.*
udp     0       0       *.route         *.*
```

The first line of data shows an "ESTABLISHED" TCP telnet connection to host "sophie". We know that we are connected to a telnet daemon because the port number of the remote socket corresponds to the well-known port number for telnet daemons. The list of well known port numbers is contained in the */etc/services* file. The local port number is 1025. This is because port numbers of client processes (such as our local *telnet*) are assigned arbitrary (and unique) port numbers. The next four lines show daemons running on our system. The reason that there is a '*' instead of the local name ("sartre") is because these sockets are "LISTEN"ing. When a connection is made to a foreign client socket, the local part of the address is filled in:

```
tcp  0   0   sartre.shell papa-ec0.1027   ESTABLISHED
```

Note that the foreign address is "papa-ec0" rather than "papa." This is because "papa" corresponds to address 200.0.4.3, while "papa-ec0" corresponds to 42.0.0.94 (the same network number as "sartre"). By using the "closest" interface, the kernel does not have to do routing inside of "papa". The last two lines show a UDP daemon process's sockets. UDP (Universal Datagram Protocol) is connectionless and stateless, therefore no "state" is shown. The process is actually the routing daemon, *routed*, which opens two UDP sockets. Only one of these port numbers has an entry in the services file (522).

The **-i** option shows all the currently configured interfaces. This is the "netstat -i" for sartre:

```
Name   Mtu    Network     Address   Ipkts  Ierrs  Opkts  Oerrs  Collis
ec0    1500   valid-engr  sartre    0      0      0      0      0
ec-rp  1500   0.0.0       0.0.0     0      0      0      0      0
ec-co  1500   0.0.0       0.0.0     0      0      0      0      0
lo0    1536   loopback    127.0.1   0      0      0      0      0
```

The first entry is the name of the interface. The second shows the Maximum Transmission Unit (MTU) size. This number is the size, in bytes, of the largest packet that can be transmitted without being segmented. The next entry shows the network number of the interface as derived from its address. In our case, the name "valid-engr" corresponds to the network number in our networks file. The next entry is the address of the interface using the rules described above ("Names and Addresses"). The following entries show the number of input packets, input errors, output packets, output errors, and collisions. The first line is the entry for the *ec0* interface. The next two lines show the values for RPC and for the connection manager used by RPC as transmitted over the *ec0* interface. These entries can be ignored because *netstat* interprets addresses in IP format, not RPC. The last entry is for the loopback interface, *lo0*. Because some network controllers cannot receive packets they transmit to themselves (e.g., broadcast packets), the loopback interface was created. Via software, the loopback interface send packets "back up the ladder" if the packet is destined for the local node. If the IP interface has not been configured or is otherwise down (e.g., ifconfig ec0 down), an asterisk appears next to the entry:

```
ec0* 1500 valid-engr sartre0    0    0    0    0
```

The **-r** option shows the routing tables used by the kernel (and IP) to decide which interface should be used to transmit. The following is an example of a "netstat -r" on sartre:

```
Routing tables
Destination     Gateway      Flags    Refcnt   Use     Interface
valid-engr      sartre       U        0        119     ec0
loopback        127.0.1      U        0        0       lo0
sasnet          sophie-ec0   UG       0        3288    ec0
```

```
catonet         twisted-ec0  UG    0     0     ec0
alphal          papa-ec0     UG    0     0     ec0
```

The first field describes the target; this can be either a node or a network. The second entry is the *gateway*. This is the node that a packet for the target should be sent to. The gateway is responsible for forwarding the packet to its final destination. The third field shows the flags associated with the routing table entry. 'U' means up; 'G' means gateway. The next entry is a reference count; it shows how many sockets are using that particular gateway. The next field is a use count; it shows how many times the route has been used. Finally, the last field shows the interface that packets should be sent to in order to get them to the gateway. Note once again that only IP packets use the routing tables; RPC packets are "single cable."

In summary, *netstat* is a tool that allows any user to monitor the current state of the network sub-system.

12. THE ROUTE PROGRAM

Route(8) is one of two programs that manipulate the routing tables (the other being the routing daemon, *routed*). *Route* allows manual updates to the routing tables. There are really two justifications for this: first, if the routing table gets totally out of sync for some unknown reason, *route* can be used to fix the problem; second, *route* can be used to define a *wild-card* router The following example is a command line in which I want to add to my routing table an entry in which "sophie" serves as the gateway to the "foobaz" network. Note that I use "sophie-ec0" because that interface is on my local network:

```
sartre # route add foobaz sophie-ec0 1
```

The last argument defines the value of the "hop count" for the gateway. The hop count is the cost of using the gateway. Generally, it is defined to be 1. If I am on my SCALDsystem IV, I have two choices if I wish to access a host on the main Ethernet (network 42). I can either run the routing daemon (which will expend computing resources I could more ably use) or define a default gateway that will handle all traffic that is not to my local network (200.0.1 for "poptart"). Because "sophie" is my file server, I will use it as my gateway:

```
poptart # route add default sophie 1
```

The command line specifies that the default target address (0.0.0.0) uses "sophie" as its gateway. Any address that "poptart" cannot find a gateway for is directed to "sophie." If "sophie" cannot find a gateway for it, it responds to "poptart" with an error packet. In the SCALDsystem IV *rc.network* file, there is line that initializes the default route:

```
: 'Initialize our default route'

route add default ${FILESERVER=none} 1
```

Remember that RPC does not use the routing tables for determining on which interface it sends packets.

13. THE ROUTED DAEMON

Routed(8) is the Berkeley routing daemon. Each system that is running the routing daemon listens for packets from other routing daemons. Systems that have more than one interface (e.g., "sophie" in the example above) broadcast to other nodes that they have a route from the network on one interface to the network on the other. All nodes receiving these packets update their own routing tables based on the information that they receive. Each "router" (e.g., "sophie") broadcasts every 30 seconds. If a node does not hear from a router in 3 minutes, it assumes the router is dead and deletes that gateway from its routing tables. *Routed* does have some quirks. First, *routed* works from its own image of the routing table; in effect there are two routing tables that are being kept: one by UNIX and one by *routed*. When *routed* senses a change in the topology (a router dies or a new router becomes active), it updates its copy of the table and then tells UNIX (via a system call) to update its copy. Second, at no time does *routed* actually read the UNIX routing table. Rather, it relies on being able to receive its own broadcast packets (via the loopback interface) to determine what interfaces are currently up. This also means that if someone modifies the routing table with *route*, the routing daemon will not be aware of the change nor will it correct the change if it does not reflect the actual topology of the network. Finally, *routed* does not support the "default" routing entry described above; if a system wishes to use default routing, it probably should not use *routed*. *Routed* is invoked on all SCALDsystems (except IV) in the following way:

```
: 'Start standard routing daemon'

/etc/routed
```

14. IP UTILITIES

The next sections describe the utilities that use the IP protocol. These programs were developed at Berkeley for use with the 4.2 BSD operating system. They can be classified into two groups: those that communicate with only BSD UNIX systems (homogeneous) and those that allow communication between BSD UNIX and other systems that support the TCP, UDP, and IP protocols (heterogeneous). The following programs are homogeneous: *rlogin*, *rsh*, and *rcp*. The following programs are heterogeneous: *telnet* and *ftp*.

15. THE RLOGIN AND RLOGIND PROGRAMS

The *rlogin*(1) and *rlogind*(8) programs allow a user from one system to remotely log in to another system via the network. This is sometimes called an NVT or Network Virtual Terminal. The command is as follows:

```
sartre % rlogin sophie
```

The rules governing authorization/authentication are described below ("Homogeneous Security"). A user may also wish to log in the target as someone else. In the following example, I wish to log in as **root** on sophie:

```
sartre % rlogin sophie -l root
```

The **-l** option tells *rlogin* that I do not wish to log in as myself; the next argument specifies who I wish to log in as. The Valid version of *rlogin* and *rlogind* allow a user in a SCALDsystem graphics window to transparently export his window characteristics to the remote system. This insures that window oriented programs such as *vi(1)* work correctly when the remote system is also a SCALDsystem.

16. THE RSH AND RSHD PROGRAMS

The *rsh*(1) and *rshd*(8) programs allow a user to execute a shell command on a remote machine. This is an example in which we want to find out who is on a remote system:

```
sartre %  rsh sophie who
root    maint     May 21 11:37
```

It is also possible to use quotes to send multiple shell commands:

```
sartre %  rsh sophie "cd /u0/sas; pwd"
/u0/sas
```

One thing that a user should be careful of is the use of pipes and re-direction in *rsh* commands. A good rule is that if the pipe or re-direct is **inside** of quotes, it occurs on the remote machine, otherwise it occurs on the local machine. For example

```
sartre %  rsh sophie echo boo > /tmp/foo
```

fills a file called /tmp/boo on **sartre** with the string "boo" while

```
sartre %  rsh sophie "echo boo > /tmp/foo"
```

fills a file called /tmp/boo on **sophie** with the string "boo."

17. THE RCP COMMAND

Rcp(1) allows users to copy files between systems. For instance, if I wanted to copy a file on "sophie" called /u0/tmp/frog to my local system and call it */u0/sas/super*, I would say:

```
sartre %  rcp sophie:/u0/tmp/frog  /u0/sas/super
```

Note that *rcp* uses the *rsh* daemon to invoke a remote *rcp*. This means that *rcp* uses the same security scheme as *rsh*. If I now wanted to copy the file back, I could say:

```
sartre %  rcp /u0/sas/super  sophie:/u0/tmp/frog
```

18. HOMOGENEOUS SECURITY

For each of the homogeneous programs the target server system is another BSD UNIX system. Assumptions are made about the security of the target. First, all systems' addresses must be given by their names as listed in */etc/hosts* on both the target and the client. This means a user cannot type:

```
sartre %  rlogin 42.0.0.94
```

Second, two files are consulted that determine the type of authorization that will be performed. These files are */etc/hosts.equiv* and *~/.rhosts* (where "~" is the home directory of the client user). There are two types of authentication checks: password verification or no password verification. In the case of *rsh* and *rcp*, if password verification is indicated, the process is terminated with a "Permission denied" message. The target system's */etc/hosts.equiv* file is checked if the client user is not root. Each entry has two fields. The first field is the name of a host. The second field is optional and contains a list of users. If there is no second field, all users on the named host are considered "equivalent" and are not password verified. If there is a second field, only those users in the field are considered equivalent; other users from that system must be verified. If the client user is root or the ***hosts.equiv*** file check fails, the *~/.rhosts* file is consulted. This file has the same format as the ***hosts.equiv*** file and the same algorithm is used to determine whether or not password verification is necessary.

Some examples are in order. First, we will show the */etc/hosts.equiv* on sophie:

```
sartresas
papa
papa-ec0
darth
server
```

Note that we specify both of the interface names for "papa". This is because it is possible that someone from "papa" will show up as being from "papa-ec0" depending on the interface chosen on "papa". See the example for *netstat* above. If, from "sartre", I wish to *rsh* to sophie as **sas** I type the following command:

```
sartre %  rsh sophie who
root    maint May 21 11:37
```

I did not require authorization because my name was in the ***hosts.equiv*** user's field for the "sartre" entry. If user **dumbo** on "sartre" wished to do the same command, he would fail:

```
sartre %  rsh sophie who
Permission denied.
```

because **dumbo** is not in the user's field in sophie's ***hosts.equiv*** file. Next, we will create a file on sophie called */u0/dumbo/.rhosts* (*/u0/dumbo* is **dumbo**'s home directory on sophie). This file has the following contents:

```
sartre
darth
server
hub
```

Now, if **dumbo** tries to *rsh* from "sartre" to "sophie", he will succeed because the .rhosts check will pass. Note that any user on "papa" (or "darth" or "server") **except root** would be able to pass the check in ***hosts.equiv*** and would not need password verification. Only if the machine name is in the */.rhosts* file (/ is the root home directory), will the user *root* on one system pass the check without password verification on "sophie."

19. TELNET AND FTP UTILITIES

The *telnet*(1) and *ftp*(1) utilities are described in separate documents included with this change package.

20. HETEROGENEOUS SECURITY

Security for heterogeneous programs is much more stringent than for the homogeneous programs. No assumptions are made about the target or client systems. The user must successfully complete a password verification any time he wishes to connect to another machine. This password test is made by the target machine, and the validation is made under the rules followed by the target system. Once the password check is complete, the user has the permissions of the user to whom he has logged in on the target machine.

21. VALID EXTENDED FILE SYSTEM

VALID's Extended File System (EFS) supports transparent file access of files on remote systems through a special path name. For instance, as a user on "sartre," to see the contents of the file */u0/sas/.cshrc* on "darth", type:

```
sartre % cat /net/darth/u0/sas/.cshrc
```

This tells UNIX that the file is on the network ('/net') on host "darth" ('/darth') and is called /u0/sas/.cshrc. There are several restrictions. First, it is not possible to nest references to /net:

```
sartre % cat /net/darth/net/server/.rhosts
cat: can't find /net/darth/net/server/.rhosts
```

The message does not mean that the file is not there, it is just not accessible using the /net phrase twice. Second, only opening, closing, reading, writing, and status checks (*stat*(2)) of remote files are allowed. In particular, it is not possible to append to remote files as this involves a seek. Third, it is not possible to reference most special devices (disks, tape drivers, etc.) using /net. This is because there is no mechanism to allow "blocking" of data across the network. Fourth, the target host must have an entry in the connection table. As described earlier, this entry provides a way to get the address of a specific host and guarantees that the host is alive. Fifth, unlike IP, user identification is kept by user id number rather than the user name string. This means that for any access to a target system, the user has the privileges of his user id number on the target system. Finally, a user id of 0 on the local system is converted into a different user id (default 11) on the remote system. The transaction has permissions associated with user id 11 on the remote system.

APPENDIX A
IP ADDRESSING FORMAT

ASSIGNED NUMBERS

This appendix is taken from the "File Transfer Protocol, Request for Comments (RFC)" section of the *Internet Protocol Transition Workbook* by Jon Postel and documents the currently assigned values from several series of numbers used in network protocol implementations. This document will be updated periodically, and current information can be obtained from Jon Postel. The assignment of numbers is also handled by Jon. If you are developing a protocol or application that requires the use of a link, socket, port, protocol, or network number, please contact Jon to receive a number assignment.

Jon Postel
USC- Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90291
(213) 822-1511
ARPANET mail: POSTELOISIF

Most of the protocols mentioned here are documented in the RFC series of notes. The more prominent and more generally used are documented in the Protocol Handbook prepared by the Network Information Center (WIC). Some of the items listed are undocumented. In all cases the name and mailbox of the responsible individual is indicated. In the lists that follow, a bracketed entry (e.g., [17,iii]) at the right hand margin of the pages indicates a reference for the listed protocol where the number references the document and the "iii" references the person.

ASSIGNED NETWORK NUMBERS

This list of network numbers is used in the internet address [33]. The Internet Protocol (IP) uses a 32-bit address and divides that address into a network part and a "rest" or local address part. The division takes three forms or classes.

The first type, or class a, of address has a 7-bit network number and a 24-bit local address. This allows 128 class A networks.

```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0!   NETWORK   |                Local Address                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Class A Address

The second type, or class B, of address has a 14-bit network number and a 16-bit local address. This allows 16,384 class B networks.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 0|           NETWORK         |          Local Address        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Class B Address

The third type, or class C, of address has a 21-bit network number and a 8-bit local address. This allows 2,097,152 class C networks.

```
                     1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 1 0|                    NETWORK              | Local Address |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Class C Address

One notation for internet host addresses commonly used divides the 32-bit address into four 8-bit fields and specifies the value of each field as a decimal number with the fields seperated by periods. For example, the internet address of ISIF is 010.020.000.052.

This notation is used in the listing of assigned network numbers; the class A networks have *nnn.rrr.rrr.rrr*, the class B networks have *nnn.nnn.rrr.rrr*, and the class C networks have *nnn.nnn.nnn.rrr* where *nnn* represents part or all of a network number and *rrr* represents part or all of a local address or rest field.

ASSIGNED NETWORK NUMBERS

The following tables define the assigned numbers within the three classes.

Class A Networks

Internet Address	Name	Network	Reference
000.rrr.rrr.rrr		Reserved	(JBP)
001.rrr.rrr.rrr	BBN-PR	BBN Packet Radio Newtork	(DCA2)
002.rrr.rrr.rrr	SF-PR-1	SF Packet Radio Network (1)	(JEM)
003.rrr.rrr.rrr	BBN-RCC	BBN RCC Network	(SGC)
004.rrr.rrr.rrr	SATNET	Atlantic Satellite Network	(DM11)
005.rrr.rrr.rrr	SILL-PR	Ft. Sill Packet Radio Network	(JEM)
006.rrr.rrr.rrr	SF-PR-2	SF Packet Radio Network (2)	(JEM)
007.rrr.rrr.rrr	CHAOS	MIT CHAOS Network	(MOON)
008.rrr.rrr.rrr	CLARKNET	SATNET subnet for Clarksburg	(DM11)
009.rrr.rrr.rrr	BRAGG-PR	Ft. Bragg Packet Radio Net	(JEM)
010.rrr.rrr.rrr	ARPANET	ARPANET	(17,1,VGC)
011.rrr.rrr.rrr	UCLNET	University College London	(PK)
012.rrr.rrr.rrr	CYCLADES	CYCLADES	(VGC)
013.rrr.rrr.rrr		Unassigned	(JBP)
014.rrr.rrr.rrr	TELENET	TELENET	(VGC)
015.rrr.rrr.rrr	EPSS	Bristish Post Office EPSS	(PK)
016.rrr.rrr.rrr	DATAPAC	DATAPAC	(VGC)
017.rrr.rrr.rrr	TRANSPAC	PRANSPAC	(VGC)
018.rrr.rrr.rrr	LCSNET	MIT LCS Network	(43,10,DDC2)
019.rrr.rrr.rrr	TYMNET	TYMNET	(VGC)
020.rrr.rrr.rrr	DC-PR	D.C. Packet Radio Network	(VGC)
021.rrr.rrr.rrr	EDN	DCEC EDN	(EC5)
022.rrr.rrr.rrr	DIALNET	DIALNET	(26,16,MRC)
023.rrr.rrr.rrr	MITRE	MITRE Cablenet	(44,APS)
024.rrr.rrr.rrr	BBN-LOCAL	BBN Local Network	(SGC)
025.rrr.rrr.rrr	RSRE-PPSN	RSRE PPSN	(BD2)
026.rrr.rrr.rrr	AUTODIN-II	AUTODIN II	(EC5)
027.rrr.rrr.rrr	NOSC-LCCN	NOSC LCCN	(KTP)
028.rrr.rrr.rrr	WIDEBAND	Wide Band Satellite Network	(CJW2)
029.rrr.rrr.rrr	DCN-COMSAT	COMSAT Dist. Comp. Network	(DLM1)
030.rrr.rrr.rrr	DCN-UCL	UCL Dist. Comp. Network	(PK)
031.rrr.rrr.rrr	BBN-SAT-TEST	BBN SATNET Test Network	(DM11)
032.rrr.rrr.rrr	UCL-CR1	UCL Cambridge Ring 1	(PK)
033.rrr.rrr.rrr	UCL-CR2	UCL Cambridge Ring 2	(PK)
034.rrr.rrr.rrr	MATNET	Mobile Access Terminal Net	(DM11)
035.rrr.rrr.rrr	NULL	UCL RSRE Null Network	(BD2)
036.rrr.rrr.rrr	SU-NET	Stanford University Ethernet	(MRC)
037.rrr.rrr.rrr	DECNET	Digital Equipment Network	(DRL)
038.rrr.rrr.rrr	DEC-TEST	Test Digital Equipment Net	(DRL)
039.rrr.rrr.rrr	SRINET	SRI Local Network	(GEOF)
040.rrr.rrr.rrr	CISLNET	CISL Multics Network	(CH2)
041.rrr.rrr.rrr	BBN-LN-TEST	BBN Local Network Testbed	(KTP)
042.rrr.rrr.rrr	SINET	LLL-S1-NET	(EAK)
043.rrr.rrr.rrr	INTELPOST	COMSAT INTELPOST	(DLM1)
044.rrr.rrr.rrr	AMPRNET	Amature Radio Experiment Net	(HM)
044.rrr.rrr.rrr- 126.rrr.rrr.rrr		Unassigned	(JBP)
127.rrr.rrr.rrr		Reserved	(JBP)

Class B Networks

Internet Address	Name	Network	References
128.000.rrr.rrr		Reserved	(JBP)
128.001.rrr.rrr- 128.254.rrr.rrr		Unassigned	(JBP)
191.255.rrr.rrr		Reserved	(JBP)

Class C Networks

Internet Address	Name	Network	References
192.000.001.rrr		Reserved	(JBP)
192.000.001.rrr- 223.255.254.rrr		Unassigned	(JBP)
223.255.255.rrr		Reserved	(JBP)

Other Reserved Internet Addresses

Internet Address	Name	Network	References
224.000.000.000- 255.255.255		Reserved	(JBP)

DESCRIPTION AND TUTORIAL ON FTP

This document is a high level overview of the SCALDsystem's version of the File Transfer Protocol, called *ftp*(1). It is highly recommended that first time users of the utility read this first before running *ftp*. More experienced users may also find some of the information in this document informative.

This is the user's interface to the ARPANET standard File Transfer Protocol. The main purpose of the *ftp* program is to give heterogeneous hosts on a network a common interface to transfer files. Thus a user on a VAX running VMS can transfer a file to a SCALDsystem running UNIX through *ftp*. *Ftp* can be thought of as a superset of the UNIX utility *rcp*(1).

1. AUDIENCE AND MOTIVATION

The following is a tutorial on the commands and behavior of the *ftp* utility on the SCALDsystem and does not require that the user has any knowledge of the Internet protocols or the version of the remote ftp server on the target system.

2. STARTING THE DAEMON

Any host can connect to another that has an *ftp* server running on their system. To allow file access on the SCALDsystem system for remote users, invoke the server (or daemon) by doing the following:

1. Login as the superuser "root."

2. Make sure the server is not running already by typing

   ```
   % ps ax
   ```

   and look for a process with a command name "etc/ftpd."

3. If the process has not been started, type the command:

   ```
   % /etc/ftpd
   ```

The server will now run in the background and will listen for connection requests from remote users on other hosts.

NOTE

Make sure the hosts have entries in the file */etc/hosts*.

3. GETTING STARTED

To start an *ftp* session, a user types the command:

```
% ftp
```

The following prompt will be displayed:

```
ftp>
```

The user is now in the *ftp* command shell. All acceptable commands can be printed out by typing **help** or '**?**.' A one line summary of the commands is printed out by typing **help** *command.* As an example, to get more infomation on the **get** command, one would type:

```
ftp> help get
```

To connect to a host (i.e., sesame) that supports *ftp,* type:

```
ftp> open sesame
```

Notice that several messages will be displayed. These messages are coming from the server on the remote host. The server will ask for a user name and a password to validate permissions for file access. If successful, a message such as

```
230 User logged in, proceed
```

will appear. If a mistake was made in typing the above, the message

```
530 Not logged in
```

comes up. If the latter is the case, the **user** command should be executed.

```
ftp> user dirk
```

The above command starts the validation sequence again and prompts user "dirk" for his password again, if it exists. All users on the system should be urged to have passwords for security reasons.

Note that there are numbers in front of the messages. The user should not be concerned about what these numbers mean. These numbers describe the state that the deamon is in and are well defined in the document *File Transfer Protocol*†.

† Postel, J., "File Transfer Protocol," *Internet Protocol Transition Workbook,* SRI Network Information Center, RFC 765, March 1982.

4. TRANSFERING FILES

After the user logs in, he can access any file that the owner of the account can. To get a listing of the current directory, a user can type in the following command:

ftp> **ls**

Some remote servers may also print out a more extended listing. Users can see this by typing in:

ftp> **dir**

The server on the SCALDsystem will print out the equivilent output as the UNIX command **ls -lg**. Refer to *ls(1)* in the *SCALDsystem UNIX Programmer's Manual* for a more complete description of the above command.

To copy a file from the remote host to his local system, a user must know whether or not the contents of the remote file are in ASCII or binary. When *ftp* is first invoked, file transfers are all assumed to be in ASCII. If binary file transfers are needed, the user can switch the transfer type by:

ftp> **binary**

To switch back to ASCII type, type in the command

ftp> **ascii**

The file can now be transfered via the **get** command. The syntax of the command is:

ftp> **get** *remote-file-name* [*local-file-name*]

Note that the local file name is optional. The default local name is the remote name. If the remote name is not acceptable, *ftp* prompts for a local file name. Once the message

226 Closing data connection; requested file action successful

is displayed, the file is on the local system.

File name syntax depends on the remote machine. An example would be VMS syntax:

drc3:[dirk.junk]foobar.c

Thus, to get the file "foobar.c" from a VMS host:

ftp> **get drc3:[dirk.junk]foobar.c foobar.c**

Ftp also allows file transfers from local hosts to remote hosts. The command going the other direction is *put,* with the syntax:

ftp> **put** *local-file-name* [*remote-file-name*]

Note that the remote file name is optional. The default here is the local file name.

There are times when a user wants to transfer more than one file over. Sometimes this can be made more convenient if the user uses the **mget** and **mput** commands. The syntax for these commands is:

```
ftp> mget remote-files
ftp> mput local-files
```

Note that **mget** does not allow the user to specify the local names and **mput** the remote file names. The user must be aware of this. The advantage of this syntax is to allow the user to name more than one specified file to be sent in one command line. Metacharacters are processed by default in *ftp*. The metacharacters that *ftp* understands are:

*	Match zero to all characters
?	Match one or zero characters
[ ]	Match any of the character patterns between the brackets
~	Home directory to be substituted here

It should be noted that some remote servers do not process metacharacters. To turn off metacharacter processing, type in the command:

```
ftp> glob
```

Because of the syntax of **mget** and **mput**, the user would need to be able to have the ability to change to the target directory. To change to the remote target directory, the user should use the command **cd**. The syntax of the remote directory depends mainly on the remote server. Some accept the UNIX notation. Refer to the users manual for the server. To change the local directory, the **lcd** command is used.

Files can also be deleted using the command

```
ftp> delete remote-file-name
```

As one would expect, there is also a command **mdelete** to remove multiple sets of files.

If the user has very large files and he wants to have assurance that *ftp* has not gotten hung up, he could use the command **hash**. To invoke "hashing":

```
ftp> hash
```

This will have *ftp* print a '#' for each 512 byte buffer transferred.

5. DISCONNECTING FROM THE REMOTE HOST

To connect to another host with an *ftp* server, the user must close the current connection by typing:

```
ftp> close
```

The user can now open another connection with another host. To end the *ftp* session, type:

```
ftp> bye
```

These commands are also described in *ftp*(1C) in the *SCALDsystem UNIX Programmer's Manual*. Some of these commands are UNIX specific and are not part of the ARPANET protocol, such as *pwd*, *mkdir*, and *rmdir*. Users should be aware of this.

6. ADVANCED ISSUES IN FTP

The next few subjects are more advanced issues regarding the *ftp* client and server. Users should feel comfortable with the above commands before reading on.

6.1. COMMUNICATION WITH THE SHELL

Another point to be made is communication with the shell and ftp. Any shell command can be executed if a '!' is typed before the command. Thus to cat a file, type:

ftp> **!cat** *local-file-name*

To print a remote file to standard output, type:

ftp> **get** *remote-file-name* **-**

Note the dash where the local file name would have been. This is a signal to *ftp* that the user needs look at the remote file. This is equivalent to a remote *cat.* The same applies to **put**, except that standard input (i.e., the keyboard) is the case. If the command is:

ftp> **put -** *remote-file-name*

the behavior acts just like the shell command line

% **cat** > *remote-file-name*

If the first character of the file name is '|', the rest of the file name is considered to be a shell command. Thus, if someone would like to have the listing of the current directory sent over to the remote host, the command would be:

ftp> **put |ls** *remote-file-name*

A useful example of this command is

ftp> **ls . |more**

if the number of remote files is greater than the number of lines on your terminal. If there are spaces that are needed to delimit the command from its arguments, quotes must be put around the shell command, such as

ftp> **dir . "|sort +4nr"**

6.2. FTP COMMANDS AT THE LOWEST LEVEL

The command **remotehelp** will return commands supported by the remote server. These are actual commands passed between client and server over the network. They are defined in the "File Transfer Protocol." Some of these commands are not be supported by the SCALDsystem *ftp* client. These commands are

REIN	PASV	MLFL	MAIL	MSND	MSOM
MSAM	MRSQ	MRCP	REST	ABOR	SITE
STAT					

The "SITE" command may be neccessary for file transfers on some systems. The file structure and type are some of the examples the "SITE" command may have an effect on. Check the remote host's server manual for a more complete description about what this command does. To send this command over, type the command

```
ftp> quote site arg1 arg2 ...
```

The command **quote** sends its arguments verbatim to the server. There must be only one return reply from the server to keep communication between the client and server in a known state.

6.3. ANONYMOUS LOGINS

The *ftp* server allows anonymous logins to the system. The system adminstrators must follow the guidelines at the end of the *ftpd*(8C) manual page for security reasons. This feature is useful to allow users without an account on the local host to access files they would not have gotten otherwise. However, the administrator must be aware that there still may be security holes in this feature.

DESCRIPTION AND TUTORIAL ON TELNET

This document is a high level overview of the SCALDsystem's version of Telnet, called *"telnet."* It is highly recommended that first-time users of the utility read this first before running *telnet*(1). More experienced users may also find some of the information in this document informative.

This document is a descrition of the user's interface to the ARPANET standard Telnet protocol. The main purpose of the *telnet* program is to give heterogeneous hosts on a network a common interface to run a login session. Thus a user on a SCALDsystem running UNIX can log on to a VAX running VMS. *Telnet* can be thought of as a superset of the UNIX utility *rlogin*(1C).

The following is a tutorial on the commands and behavior of the *telnet* utility on the SCALD-system and does not require that the user have any knowledge of the Internet protocols, but some level of expertise about the target system's operating system is neccessary.

1. STARTING THE DAEMON

Any user on a host can log on to a remote host if that remote host has a telnet server running. To allow remote users to log on to the SCALDsystem via *telnet*, start the server (or daemon) by following these steps:

1. Login as the superuser "root."
2. Make sure the server is not running already by typing

   ```
   % ps ax
   ```

   and look for a process with a command name "etc/telnetd."
3. If the process has not been started, type the command:

   ```
   % /etc/telnetd
   ```

   The telnet server will now run in the background and listen for connection requests from remote users on other hosts.

NOTE

Make sure the remote hosts have entries in the file */etc/hosts.*

2. GETTING STARTED

To start a telnet session, a user types the command:

```
% telnet
```

The following prompt will be displayed:

```
telnet>
```

The user is now in the *telnet* command shell. All acceptable commands can be printed out by typing '**?** .' One line command summaries are printed out.

To log on to a remote host (i.e., sesame), type the following:

telnet> **open sesame**

A login prompt unique to the host will be displayed. Log on as if you were on a terminal connected to the host.

3. RETURNING TO THE TELNET COMMAND SHELL

Note that the escape character is 'CNTL-]' (printed out as '^]'). If a user types in this character, he is returned to the *telnet* command shell.

If the remote system takes 'CNTL-]' to be special character, the user can change the escape character by the *telnet* command **escape**. For example, to change the escape character to 'CNTL-x':

telnet> **escape**

Telnet will prompt for a character to become the new escape character. Type in CNTL-x. *Telnet* sets the new escape and returns the user to the login session.

4. DISCONNECTING FROM REMOTE HOST

If the user would like to log on to another system, he would type in the escape character and type in:

telnet> **close**

Telnet will automatically log him off the remote machine, and the user can now open another host on which to log on.

To quit the *telnet* session, the user can do one of two things:

1. He can log off the remote system

2. He can type in the escape character and enter:

   telnet> **quit**

UUCP OPERATION AND MAINTANENCE

1. INTRODUCTION

This document describes the setup, operation and maintenance of Valid's 8.0 release of the **uucp** communications package on the S-32 computer system. The document is not intended to be a complete description of the uucp package, but rather an aid in bringing up *uucp* on the S-32. For a more complete description of the uucp package, see *Uucp Implementation Description* by D. A. Nowitz. This supplemental document is also included in the Valid change package for release 8.0.

2. OVERVIEW

2.1. FILES

The *uucp* package consists of several program, data, and spool directory files that are spread out between the following directories:

/usr/bin	Contains program files which are executed by the user.
/usr/lib/uucp	Contains program and data files used by the programs in */usr/bin*.
/etc	Contains some programs used for uucp maintenance.
/usr/spool	Contains the spool and working directories for uucp.

The pertinent files consist of four programs:

/usr/bin/uucp
/usr/bin/uux
/usr/lib/uucico
/usr/lib/uuxqt

four data files (all in */usr/lib/uucp*) :

L-devices
L.sys
USERFILE
L.cmds

and the relevant documentation located in */usr/doc/uucp*.

Uucp is a spooler for file copying, and *uux* is the spooler for remote command execution. *Uucico* and *uuxqt* do the dirty work of actually copying files and executing commands spooled by *uucp* and *uux*. The data files contain identification/consistency information that is used by *uucp*. It is strongly suggested that the document, *Uucp Implementation Description* be read BEFORE attempting to configure your uucp site. This document is located in the */usr/doc/uucp* directory and can be printed off via **nroff -ms implement | lpr**.

2.2. OPERATION

A short overview of uucp's operation is in order to provide a little motivation to the following installation instructions.

A user, say, sara, on the obiwan system, wishes to copy a data file to her account on the dagobah UNIX system. She types

```
% uucp compile.log dagobah!/u0/sara/compile.log
```

(The % is her prompt.) Uucp checks some permissions and copies the file *compile.log* into a standard spool directory */usr/spool/uucp*. It then executes *uucico.* Uucico reads the data files to determine whether it can call dagobah, and the appropriate device. It takes the point of a person approaching a terminal: it reads the "login:" message, types a user-name (in this case, uucp), reads the "Password:" message, and types its password. When all this is done, the user *uucp* is logged onto the remote system.

The shell for the uucp login process is not the standard shell, but uucico. This uucico establishes a dialogue with the uucico on the initiating system and the two uucico's are now free to copy files back and forth to each other. The uucico on obiwan reads sara's file out of its spool file into its tty. On the other end, the dagobah uucico reads the file from its input and copies it into */u0/sara/compile.log.* And that's it. If the transfer was unsuccessful, sara (on obiwan) will get mail back saying so.

The *uux* command works in same way. Let's say sara wishes to print a file using the printer on a remote system, say "remsys." Let's also say that her system has only a uucp connection to remsys (i.e., she cannot use *rsh* or *rlogin* to get to "remsys"). She could print her file, call it "file," as follows:

```
% cat file | uux - remsys!lpr
```

This command passes *file* to the uux command, (the '-' tells uux to take it's input from "stdin"). Uux then calls uucico to transfer the file to remsys along with some instructions on what to do with it once it gets there. When the file arrives on remsys, the uuxqt program there directs *lpr* to print the file.

3. INSTALLATION PROCEDURE

3.1. GETTING STARTED

The Valid 8.0 UNIX release contains all files needed to configure your S-32 as a uucp site. All uucp programs and data files are included in the release and all needed directories are created during the installation. The data files contain example entries to help in configuring your particular site.

3.2. HARDWARE SETUP

Uucp is made to work over an RS-232 direct connect or RS-232 to ACU (automatic call unit modem). The following hardware is needed to set up a uucp connection:

S-32 computer with UNIX Terminal Board
Straight through male to male RS-232 cable (9-pin if ACU being used)
Null modem connector (pins 2 and 3 swapped)
Automatic Call Unit Modem (Optional)

3.2.1. Direct Connect

If you wish to talk to another uucp system via modem skip to the section entitled "ACU Setup." If you desire a direct connection between two S-32's or an S-32 and another UNIX system, do the following:

1. Connect the RS-232 cable together with the null modem connector.
2. Connect one end of the cable and null modem connector to a UTB port on the S-32.
3. Connect the other end of the cable to an RS-232 port on the other UNIX system (or a UTB port on the other S-32).

Skip to the section entitled "Site Name Selection."

3.2.2. ACU Setup

If you have another UNIX system available, it is suggested that you first test your uucp configuration with a direct connection. When your uucp connection is working, you may set up your S-32 for connection to an ACU as follows:

1. Connect the RS-232 cable together with the null modem connector.
2. Connect one end of the cable and null modem connector to a UTB port on the S-32.
3. Connect the other end of the cable to the RS-232 port on the modem.
4. Set the modem for Auto Dial (or Auto Answer if you want another site to call your site)

You may need to set the modem for High or Low speed depending on the baud rate at which you wish uucp to run.

NOTE: The UNIX Terminal Board in the S-32 is a "dumb" octal serial board (i.e., the bulk of the I/O processing is actually done by the cpu). For this reason it is recommended that the baud rate on the modem and port be set to 300 for a moderate to heavily loaded system or if large files are to be transferred.

3.3. TTYS

After connecting the hardware, your ttys file, */etc/ttys*, must be edited. Determine the device file names of the terminals you just connected together (i.e., J0 <-> /dev/ttya, J1 <-> /dev/ttyb, etc).

NOTE: J0 is the name of the RS-232 port as printed on the backplane and /dev/ttya is it's name in the ttys file.

On each system connected, you must edit the ttys file to contain correct information for the terminals. If, for example, you connected to port ttyd on the S-32, you would change the entry for ttyd so that it contained

```
15ttyd
```

The '1' indicates that the terminal is active, i.e., people (or uucp) can log in on it. The '5' is a code meaningful to *getty*(8) (/etc/getty), the program that controls the log-in process. In this case, '5' means uucp - 300 baud. (If you wish to use 1200 baud, specify '3' instead of '5')

If you intend to call out on the port you just connected, the port must be disabled by putting a 0, instead of a 1, in the first character position of the entry. You must also change the permissions of the device itself so that uucp can access it. Do this by using the *chmod*(1) command. For example, if ttyd is the port I want to use:

```
chmod 666 /dev/ttyd
```

Before you can use uucp, you must execute the following command:

```
kill -1 1
```

This command causes *init*(8) to reread the ttys file and make the changes you just made take effect.

3.4. SITE NAME SELECTION

Uucp uses the host name of the S-32 as it's site (node) name. The host name is derived from the string "MACHNAME" in the */etc/configuration* file. This may cause problems since uucp only recognizes a site name of seven characters or less (Bell System V UNIX only recognizes six characters). This is a throw back to the early days of uucp, but we are stuck with it since so many uucp sites are running older versions of uucp. For this reason it is recommended that you pick a site name that uniquely identifies your company name in seven or less characters to avoid much of the confusion in conversing with other companies on a uucp network (one such network is described in the document entitled "network" in */usr/doc/uucp*).

To change the host name of your S-32 just change the "MACHNAME" string in */etc/configuration* to the new host name and reboot the system.

3.5. CREATING THE UUCP LOGIN

The next step in the installation procedure is to add the uucp login name to the password file, */etc/passwd.*

NOTE: The default passwd file in Release 8.0 already contains this user name. If 8.0 is the first release of software you have ever received from Valid, you can skip this step.

The uucp login is a "remote" login that has limited privileges — it doesn't get a regular shell, but rather is only allowed to copy and receive files from remote systems.

When editing the password file, you need to specify the following information for the uucp user:

Name: uucp
Uid: (whatever you like, must be unique)
Gid: 0
Real Name: Uucp Remote Login
Login directory: /usr/spool/uucppublic
Login shell: /usr/lib/uucp/uucico

The following sample entry may be used:

```
uucp::20:0:Uucp Remote Login:/usr/spool/uucppublic:/usr/spool/uucp/uucico
```

Use the */bin/passwd* program to create a password for uucp. The uucp password will be needed by a remote uucp site to login to your site.

4. DATA FILES USED BY UUCP

The following data files are used by uucp: *L.sys, L-devices, USERFILE,* and *L-dialcodes.*

4.1. L.sys

The file */usr/lib/uucp/L.sys* has the format:

```
sys-name calltimes device-tag speed phone-number login-info
```

This is the set of instructions for calling the system "sys-name." Calltimes are the hours during the day in which it's OK to call; device-tag is the type of device you are using (i.e., ACU for modem or DIR for direct connect). speed is the line baud rate; phone-number is the phone number of the remote system (if you have a hardwired device, use the name of the tty here; see example below). The rest of the line is expect/send pairs. Once the phone connection has been made, we wait for the remote system to send the "expect" string. If we don't get it, we time out — the call failed. If we do get it, then we send the "send" string.

Examples:

```
        dagobah Never DIR 1200 ttyd "" "" login--login uucp ssword: hairpin

        valid Any ACU 300 5551212 "" "" ogin:--ogin:--ogin: uucp ssword: fatchance

        foobar Any tty01 1200 tty01 "" PAUSE5 "" Q\c "" \
             \012c "" \015\c "" \012\c "" \015\c BER?... \
             4155551212\015\c MODE. PAUSE5 \
             ogin: --ogin: uucp ssword: toerail
```

System "dagobah" can be called at any time, 1200 baud, on ttyd; System "valid" can be called through a modem at 300 baud. (The actual tty port that ACU is connected to is specified in the *L-devices* file.) The last example shows how the *L.sys* file can be used to specify the dial sequence of a modem unknown to uucp (in this case, a Cermatek modem). A list of modems supported by uucp along with a list of modem control characters are listed in the appendix.) The "" "" login--login gibberish is an idiom used to establish a connection with the remote system. If it doesn't send a login message in the span of a time-out period, we send it a BREAK, which is a signal to try again. We read the next login message and proceed with logging on.

4.2. L-devices

The *L-devices* file defines uucp's access to the hardware devices it will use. The format for each entry is

```
        type line dial-device speed dial-type
```

A typical *L-devices* file may contain for example:

```
     DIR ttyd unused 1200 direct
     ACU ttya unused 300 va212
```

The first line says that I have a DIRect connection on ttyd at 1200 baud. The second line indicates that I have a modem attached to port ttya and it will run at 300 baud. The "va212" in the dial-type field tells uucp to use the dial routine for a Racal Vadic 212 modem to talk to this ACU. (Appendix B contains a list of modem dial routines currently supported by uucp.) The call-unit field is only used for a DEC "dn11" autocall unit interface and, for our purposes, this field is always "unused."

4.3. USERFILE

The file */usr/lib/uucp/USERFILE* explains which users can extract files from the system. It contains lines of the form

```
        username,sysname pathname ...
```

This file is searched for the name of the current user on the current system: the first line that contains a member of this pair is allowed to copy the associated files. Username and sysname fields may be empty, in which case they stand for "any." Examples:

```
sara,obiwan /u0/lib /u0/sara
sbs, /u0/lib
```

In this USERFILE, remote user sara, on system obiwan, can copy files in */u0/sara* or in */u0/lib*. User sbs logged in on any system can copy files from */u0/lib*.

```
,marketing /u0/lib
, /u0/lib/lsttl
```

In this USERFILE, any user on the marketing machine can copy from */u0/lib*; users on other machines must be content with copying from */u0/lib/lsttl*.

If you are not running a tightly-restricted site you may want to use

```
, /
```

which means "any user, on any system, may copy any file."

4.4. L.cmds

The *L.cmds* file specifies which commands a remote system can execute via *uux* on your system. An example *L.cmds* file might look like:

```
PATH=/bin:/usr/bin:/usr/ucb
lpr
who
rmail
```

This file says that a user on a remote machine can execute the commands *lpr, who,* and *rmail* on my system. The line containing "PATH" tells the remote uux where to look for these commands on my system.

NOTE: The *rmail* command is used to handle mail sent from a user on the remote machine to a user on my machine. It is actually just a restricted version of *mail(1)*.

5. MAINTENANCE

The uucp and uux commands spool their work files into the */usr/spool/uucp* directory. This directory also contains several other files used by uucp and uux in their normal operation. The relevant files are:

LOGFILE	Contains information about spooled requests, calls to remote systems, execution of uux commands and file copy results.
SYSLOG	Contains data transfer statistics.
STST.sysname	These files are created by *uucico* and contain information on failures such as login, dialup or sequence check. The "sysname" part is composed of the name of the remote system and a sequence number obtained from the sequence check file *SEQN* in the */usr/lib/uucp* directory.

LCK..str	Lock files are created for each device in use (i.e., tty device) and each system conversing. This prevents attempts to call up a system that is already talking to our system. The "str" part is either the name of the system or device in use.
TM.pid.ddd	When a file is being copied from a remote site, it is written into a temporary file by this name. After the entire remote file is received, the "TM" file is moved to the requested destination. If processing is abnormally terminated the "TM" file remains in the spool directory. The "pid" and "ddd" fields in the name are the process ID and sequence number of the file.
AUDIT	This file is created when running uucp in debug mode. Debug mode is described in the section entitled "Trouble Shooting."
XTMP	This is a directory which is used by uuxqt to hold temporary files.
OLD	This is a directory used by the maintenance scripts to store old logfiles.

There are several scripts/programs included in the release to make maintenance of the uucp system easier. They all look at the files listed above and extract certain useful information from them. The group of programs used for maintenance include: *uuclean, uulog, uurate, uusage*, and *uusnap*. Their use is described in the SCALDsystem UNIX Programmer's Manual.

The scripts *uucp.hourly* and *uucp.daily* are provided to automate the administration process. They call the various programs listed above to produce accounting reports and/or clean up the spool directories. These scripts are meant to be run from the *crontab* file in the wee hours of the morning.

6. SECURITY

NOTE

The uucp system is a potential security loophole if not properly administered.

The uucp package in the 8.0 release has permissions set for a fairly restricted site. *Appendix A* contains a list of all the files in uucp along with their suggested permission settings. You may wish to relax some of the security, but be aware of the following considerations before changing any permissions:

- The *uucp, uux, uucico*, and *uuxqt* programs MUST be setuid to uucp to operate correctly.
- The *L.sys* file should never be readable by the casual user since it contains phone numbers and passwords for remote systems.
- The following spool directories should be set to mode 777 (read/write/search by anyone) to avoid spooling problems.

  /usr/spool/uucp
  /usr/spool/uucp/XTMP
  /usr/spool/uucppublic

You may wish to experiment to find the best balance between ease of use and security for your site.

7. TROUBLE SHOOTING

The easiest way to test the uucp system once the hardware is connected is to run uucp in it's debug mode. The **-x** option followed by a level number from 1 to 9 gives debugging information for that level. Basically, level 9 gives much more debug information than level 1. When running in debug mode, uucp creates a file called "AUDIT" on both the local and the remote machine. These files contain a running dialogue of the uucico connections as seen from each end. These AUDIT files are very helpful in troubleshooting if the problem isn't apparent.

If you have a second S-32 or another UNIX system available, set up a direct connection to it for testing. A direct connection allows you to see and control what happens at both ends of the connection.

A typical scenario might be two systems ("sysl" and "sys2") with a direct connection between them. Assuming that the file "syslfile" is to be transferred from sysl to sys2, the first step is to test the spooling of "sysfile":

```
% uucp -r -x9 syslfile sys2!/tmp
```

The **-r** option tells uucp to spool the file, but not to start the *uucico* program to process it. If everything went well you should see a "C.sys2seqnum" and a "D.sys2seqnum" file in the */usr/spool/uucp* directory (the 'C.' file contains instructions for uucico and the 'D.' file contains the contents of syslfile). If not, the debugging statements should give you a clue as to the problem. At this stage, the problems are usually always permission problems. If you suspect a permissions problem, check the perms on "syslfile" (it must be readable by the user uucp; i.e., 444) and the permissions on the spool directory. Also make sure that uucp and uucico have permissions 6111 and are owned by uucp.

The next step to try to make contact with the remote system and make the actual file transfer:

```
% /usr/lib/uucp/uucico -r1 -ssys2 -x9
```

In this test we start up the uucico program with the **-r**, **-s**, and **-x** options. The **-r** option with a '1' tells uucico to run in master mode (i.e., this tells uucico that it will have to make the first contact with the other system). The **-s** option tells uucico that we just want to contact the system called "sys2" (normally uucico looks through the spool queue and trys to contact any system that for which it has work). The **-x** option, again, tells uucico to display debugging information. The debug information from uucico is fairly cryptic, but the main sequence of events is as follows:

1. Sysl's uucico accesses it's own tty (or ACU) device to call out.
2. Sysl's uucico gets a login prompt from sys2 and logs in as uucp.
3. The two uucicos do some handshaking and choose a protocol to use.
4. Sysl's uucico tells sys2's uucico that it is going transfer a file.
5. Sys2's uucico acknowledges the file transfer and data is transmitted.

6. After the file is transferred, sys1's uucico asks sys2's uucico if it has any files to transfer back to sys1.

7. If there are no more files to transfer, the sys1 and sys2 uucicos agree to hangup, and the connection is terminated.

During the data transfer, the debug mode displays statements like:

```
send 222
rec h->cntl 42
state - 10
```

The "send" and "rec" statements show that a control packet was sent and that an acknowledgement was received from the remote computer. This is an error checking scheme to ensure that any data packets not received by the remote computer are retransmitted. This type of scheme is used quite frequently for ensuring data transfers over unreliable media such as phone lines. An "alarm" statement appearing during a debug run indicates that an acknowlegment of a data packet was not received. If you are getting many "alarm" statements, try lowering the baud rate. The "state" statement indicates the state of the data line. The '10' indicates that the line is alive and functioning.

One of the problems to watch for at this stage is to be sure that the device names and baud rates for the line being used are the same in the *L.sys* and *L-devices* files.

If you try to rerun the *uucico* command and you get the message: "Retry time not reached," you must remove the "STST." file in the spool directory. This file is left by uucico after an aborted run and is used to keep uucico from trying to repeatedly contact a remote system that is down. Before removing the "STST." file you should look at it's contents to see what uucico was doing when it failed.

As mentioned above, the "AUDIT" file created by uucico contains useful debugging info at this step. Besides "AUDIT" and "STST" files, the "LOGFILE" and "SYSLOG" files are also useful debugging tools.

We have found that about ninety percent of the problems encountered at the uucico stage are caused by incorrect entries in the *L.sys* and *L-devices* files or by incorrect hookup of the RS-232 line or modem.

APPENDIX A
UUCP FILES

The following is a list of all of the files that comprise the uucp package along with their correct permissions and a brief description. For a more detailed description please refer to the *SCALD-system UNIX Programmer's Manual*

File	Perms/Owner	Description
/etc/uucp.daily	755 uucp	Maintenance script
/etc/uucp.hourly	755 uucp	Maintenance script
/etc/uurate	755 uucp	Report uucp transfer rate
/etc/uuusage	755 uucp	Report uucp user usage
/usr/bin/uucp	6111 root	Unix to unix copy program
/usr/bin/uudecode	755 root	Encode binary file for transfer
/usr/bin/uuencode	755 root	Decode binary file for transfer
/usr/bin/uulog	755 root	Summarize uucp logfiles
/usr/bin/uuname	755 root	Return list of remote uucp sites
/usr/bin/uupoll	755 root	Poll remote uucp sites
/usr/bin/uuq	755 uucp	Report spooled files for each site
/usr/bin/uusend	755 root	Send a file to a remote host
/usr/bin/uusnap	755 root	Show snapshot of UUCP system
/usr/bin/uux	6111 root	Unix to unix command execution
/usr/lib/uucp/L-devices	444 uucp	Uucp devices data file
/usr/lib/uucp/L-dialcodes	444 uucp	Telephone area code data file
/usr/lib/uucp/L.cmds	444 uucp	Command execution data
/usr/lib/uucp/L.sys	400 uucp	Uucp dial information data file
/usr/lib/uucp/USERFILE	400 uucp	File access data file
/usr/lib/uucp/uucico	6111 root	Uucp spool file processing program
/usr/lib/uucp/uuclean	755 root	Uucp spool queue cleanup program
/usr/lib/uucp/uuxqt	6111 root	Uucp spool file processing program

APPENDIX B
SUPPORTED MODEMS

Uucp knows how to talk to the following modems. The first name is the name used in the *L-devices* file. If you wish to use a modem not listed here, you must write your own dailer routine and insert it into the *L.sys* file. (See the section describing *L.sys* for more information.)

dn11	Standard DEC dialer
hayes	Hayes' Smartmodem
ventel	Ventel dialer
vadic	Racal-Vadic 3450
va212	Racal-Vadic VA212 autodial
va811s	Racal-Vadic 811s
rvmacs	Racal-Vadic MACS 831

APPENDIX C
SPECIAL CHARACTERS

The following are recognized "sendthem" characters and their meanings. These special characters are used to write dailer routines for modems not known to uucp. (See example entry in the "L.sys" section.)

BREAKn	real break, or n nulls
EOT	\04
PAUSEn	sleep(n), default 3
LF	line feed (same as \n\c)
CR	carriage return (same as \r\c)
\ooo	byte represented by octal ooo
\c	suppress ending '\r'
\n	newline
\s	space
\d	delay (same as sleep(1))
\r	return (same as \015)
\bn	same as BREAKn
\\\\	'\\'
P_ZERO	change to space parity
P_ONE	change to mark parity
P_EVEN	change to even parity
P_ODD	change to odd parity

Lines ending with '\' are continued on the next line. Lines starting with '#' are ignored as comments and may be continued.

MAIL REFERENCE MANUAL

Kurt Shoens

Revised by

Craig Leres

Version 2.18

July 27, 1983

1. Introduction

Mail provides a simple and friendly environment for sending and receiving mail. It divides incoming mail into its constituent messages and allows the user to deal with them in any order. In addition, it provides a set of *ed*-like commands for manipulating messages and sending mail. *Mail* offers the user simple editing capabilities to ease the composition of outgoing messages, as well as providing the ability to define and send to names which address groups of users. Finally, *Mail* is able to send and receive messages across such networks as the ARPANET, UUCP, and Berkeley network.

This document describes how to use the *Mail* program to send and receive messages. The reader is not assumed to be familiar with other message handling systems, but should be familiar with the UNIX[1] shell, the text editor, and some of the common UNIX commands. "The UNIX Programmer's Manual," "An Introduction to Csh," and "Text Editing with Ex and Vi" can be consulted for more information on these topics.

Here is how messages are handled: the mail system accepts incoming *messages* for you from other people and collects them in a file, called your *system mailbox*. When you login, the system notifies you if there are any messages waiting in your system mailbox. If you are a *csh* user, you will be notified when new mail arrives if you inform the shell of the location of your mailbox. On version 7 systems, your system mailbox is located in the directory /usr/spool/mail in a file with your login name. If your login name is "sam," then you can make *csh* notify you of new mail by including the following line in your .cshrc file:

```
set mail=/usr/spool/mail/sam
```

When you read your mail using *Mail*, it reads your system mailbox and separates that file into the individual messages that have been sent to you. You can then read, reply to, delete, or save these messages. Each message is marked with its author and the date they sent it.

[1] UNIX is a trademark of Bell Laboratories.

2. Common usage

The *Mail* command has two distinct usages, according to whether one wants to send or receive mail. Sending mail is simple: to send a message to a user whose login name is, say, "root," use the shell command:

```
% Mail root
```

then type your message. When you reach the end of the message, type an EOT (control−d) at the beginning of a line, which will cause *Mail* to echo "EOT" and return you to the Shell. When the user you sent mail to next logs in, he will receive the message:

```
You have mail.
```

to alert him to the existence of your message.

If, while you are composing the message you decide that you do not wish to send it after all, you can abort the letter with a RUBOUT. Typing a single RUBOUT causes *Mail* to print

```
(Interrupt -- one more to kill letter)
```

Typing a second RUBOUT causes *Mail* to save your partial letter on the file "dead.letter" in your home directory and abort the letter. Once you have sent mail to someone, there is no way to undo the act, so be careful.

The message your recipient reads will consist of the message you typed, preceded by a line telling who sent the message (your login name) and the date and time it was sent.

If you want to send the same message to several other people, you can list their login names on the command line. Thus,

```
% Mail sam bob john
Tuition fees are due next Friday.  Don't forget!!
<Control−d>
EOT
%
```

will send the reminder to sam, bob, and john.

If, when you log in, you see the message,

```
You have mail.
```

you can read the mail by typing simply:

```
% Mail
```

Mail will respond by typing its version number and date and then listing the messages you have waiting. Then it will type a prompt and await your command. The messages are assigned numbers starting with 1 − you refer to the messages with these numbers. *Mail* keeps tack of which messages are *new* (have been sent since you last read your mail) and *read* (have been read by you). New messages have an N next to them in the header listing and old, but unread messages have a U next to them. *Mail* keeps track of new/old and read/unread messages by putting a header field called "Status" into your messages.

To look at a specific message, use the **type** command, which may be abbreviated to simply **t**. For example, if you had the following messages:

```
N 1 root    Wed Sep 21 09:21  "Tuition fees"
N 2 sam     Tue Sep 20 22:55
```

you could examine the first message by giving the command:

```
type 1
```

which might cause *Mail* to respond with, for example:

```
Message  1:
From root  Wed Sep 21 09:21:45 1978
Subject: Tuition fees
```

```
Status: R

Tuition fees are due next Wednesday. Don't forget!!
```

Many *Mail* commands that operate on messages take a message number as an argument like the **type** command. For these commands, there is a notion of a current message. When you enter the *Mail* program, the current message is initially the first one. Thus, you can often omit the message number and use, for example,

```
t
```

to type the current message. As a further shorthand, you can type a message by simply giving its message number. Hence,

```
1
```

would type the first message.

Frequently, it is useful to read the messages in your mailbox in order, one after another. You can read the next message in *Mail* by simply typing a newline. As a special case, you can type a newline as your first command to *Mail* to type the first message.

If, after typing a message, you wish to immediately send a reply, you can do so with the **reply** command. **Reply**, like **type**, takes a message number as an argument. *Mail* then begins a message addressed to the user who sent you the message. You may then type in your letter in reply, followed by a <control-d> at the beginning of a line, as before. *Mail* will type EOT, then type the ampersand prompt to indicate its readiness to accept another command. In our example, if, after typing the first message, you wished to reply to it, you might give the command:

```
reply
```

Mail responds by typing:

```
To: root
Subject: Re: Tuition fees
```

and waiting for you to enter your letter. You are now in the message collection mode described at the beginning of this section and *Mail* will gather up your message up to a control−d. Note that it copies the subject header from the original message. This is useful in that correspondence about a particular matter will tend to retain the same subject heading, making it easy to recognize. If there are other header fields in the message, the information found will also be used. For example, if the letter had a "To:" header listing several recipients, *Mail* would arrange to send your replay to the same people as well. Similarly, if the original message contained a "Cc:" (carbon copies to) field, *Mail* would send your reply to *those* users, too. *Mail* is careful, though, not too send the message to *you*, even if you appear in the "To:" or "Cc:" field, unless you ask to be included explicitly. See section 4 for more details.

After typing in your letter, the dialog with *Mail* might look like the following:

```
reply
To: root
Subject: Tuition fees

Thanks for the reminder
EOT
&
```

The **reply** command is especially useful for sustaining extended conversations over the message system, with other "listening" users receiving copies of the conversation. The **reply** command can be abbreviated to **r**.

Sometimes you will receive a message that has been sent to several people and wish to reply *only* to the person who sent it. **Reply** with a capital **R** replies to a message, but sends a copy to the sender only.

If you wish, while reading your mail, to send a message to someone, but not as a reply to one of your messages, you can send the message directly with the **mail** command, which takes as arguments the names of the recipients you wish to send to. For example, to send a message to "frank," you would do:

```
mail frank
This is to confirm our meeting next Friday at 4.
EOT
&
```

The **mail** command can be abbreviated to **m**.

Normally, each message you receive is saved in the file *mbox* in your login directory at the time you leave *Mail.* Often, however, you will not want to save a particular message you have received because it is only of passing interest. To avoid saving a message in *mbox* you can delete it using the **delete** command. In our example,

```
delete 1
```

will prevent *Mail* from saving message 1 (from root) in *mbox.* In addition to not saving deleted messages, *Mail* will not let you type them, either. The effect is to make the message disappear altogether, along with its number. The **delete** command can be abbreviated to simply **d**.

Many features of *Mail* can be tailored to your liking with the **set** command. The **set** command has two forms, depending on whether you are setting a *binary* option or a *valued* option. Binary options are either on or off. For example, the "ask" option informs *Mail* that each time you send a message, you want it to prompt you for a subject header, to be included in the message. To set the "ask" option, you would type

```
set ask
```

Another useful *Mail* option is "hold." Unless told otherwise, *Mail* moves the messages from your system mailbox to the file *mbox* in your home directory when you leave *Mail.* If you want *Mail* to keep your letters in the system mailbox instead, you can set the "hold" option.

Valued options are values which *Mail* uses to adapt to your tastes. For example, the "SHELL" option tells *Mail* which shell you like to use, and is specified by

```
set SHELL=/bin/csh
```

for example. Note that no spaces are allowed in "SHELL=/bin/csh." A complete list of the *Mail* options appears in section 5.

Another important valued option is "crt." If you use a fast video terminal, you will find that when you print long messages, they fly by too quickly for you to read them. With the "crt" option, you can make *Mail* print any message larger than a given number of lines by sending it through the paging program *more.* For example, most CRT users should do:

```
set crt=24
```

to paginate messages that will not fit on their screens. *More* prints a screenful of information, then types --MORE--. Type a space to see the next screenful.

Another adaptation to user needs that *Mail* provides is that of *aliases.* An alias is simply a name which stands for one or more real user names. *Mail* sent to an alias is really sent to the list of real users associated with it. For example, an alias can be defined for the members of a project, so that you can send mail to the whole project by sending mail to just a single name. The **alias** command in *Mail* defines an alias. Suppose that the users in a project are named Sam, Sally, Steve, and Susan. To define an alias called "project" for them, you would use the *Mail* command:

```
alias project sam sally steve susan
```

The **alias** command can also be used to provide a convenient name for someone whose user name is inconvenient. For example, if a user named "Bob Anderson" had the login name "anderson,"" you might want to use:

```
alias bob anderson
```

so that you could send mail to the shorter name, "bob."

While the **alias** and **set** commands allow you to customize *Mail*, they have the drawback that they must be retyped each time you enter *Mail*. To make them more convenient to use, *Mail* always looks for two files when it is invoked. It first reads a system wide file "/usr/lib/Mail.rc," then a user specific file, ".mailrc," which is found in the user's home directory. The system wide file is maintained by the system administrator and contains **set** commands that are applicable to all users of the system. The ".mailrc" file is usually used by each user to set options the way he likes and define individual aliases. For example, my .mailrc file looks like this:

```
set ask nosave SHELL=/bin/csh
```

As you can see, it is possible to set many options in the same **set** command. The "nosave" option is described in section 5.

Mail aliasing is implemented at the system-wide level by the mail delivery system *sendmail*. These aliases are stored in the file /usr/lib/aliases and are accessible to all users of the system. The lines in /usr/lib/aliases are of the form:

alias: name_1, name_2, name_3

where *alias* is the mailing list name and the $\textit{name}_i$ are the members of the list. Long lists can be continued onto the next line by starting the next line with a space or tab. Remember that you must execute the shell command *newaliases* after editing /usr/lib/aliases since the delivery system uses an indexed file created by *newaliases*.

We have seen that *Mail* can be invoked with command line arguments which are people to send the message to, or with no arguments to read mail. Specifying the **−f** flag on the command line causes *Mail* to read messages from a file other than your system mailbox. For example, if you have a collection of messages in the file "letters" you can use *Mail* to read them with:

```
% Mail −f letters
```

You can use all the *Mail* commands described in this document to examine, modify, or delete messages from your "letters" file, which will be rewritten when you leave *Mail* with the **quit** command described below.

Since mail that you read is saved in the file *mbox* in your home directory by default, you can read *mbox* in your home directory by using simply

```
% Mail −f
```

Normally, messages that you examine using the **type** command are saved in the file "mbox" in your home directory if you leave *Mail* with the **quit** command described below. If you wish to retain a message in your system mailbox you can use the **preserve** command to tell *Mail* to leave it there. The **preserve** command accepts a list of message numbers, just like **type** and may be abbreviated to **pre**.

Messages in your system mailbox that you do not examine are normally retained in your system mailbox automatically. If you wish to have such a message saved in *mbox* without reading it, you may use the **mbox** command to have them so saved. For example,

```
mbox 2
```

in our example would cause the second message (from sam) to be saved in *mbox* when the **quit** command is executed. **Mbox** is also the way to direct messages to your *mbox* file if you have set the "hold" option described above. **Mbox** can be abbreviated to **mb**.

When you have perused all the messages of interest, you can leave *Mail* with the **quit** command, which saves the messages you have typed but not deleted in the file *mbox* in your login directory. Deleted messages are discarded irretrievably, and messages left untouched are preserved in your system mailbox so that you will see them the next time you type:

```
% Mail
```

The **quit** command can be abbreviated to simply **q**.

If you wish for some reason to leave *Mail* quickly without altering either your system mailbox or *mbox*, you can type the **x** command (short for **exit**), which will immediately return you to the Shell without changing anything.

If, instead, you want to execute a Shell command without leaving *Mail*, you can type the command preceded by an exclamation point, just as in the text editor. Thus, for instance:

```
!date
```

will print the current date without leaving *Mail*.

Finally, the **help** command is available to print out a brief summary of the *Mail* commands, using only the single character command abbreviations.

3. Maintaining folders

Mail includes a simple facility for maintaining groups of messages together in folders. This section describes this facility.

To use the folder facility, you must tell *Mail* where you wish to keep your folders. Each folder of messages will be a single file. For convenience, all of your folders are kept in a single directory of your choosing. To tell *Mail* where your folder directory is, put a line of the form

```
set folder=letters
```

in your *.mailrc* file. If, as in the example above, your folder directory does not begin with a '/,' *Mail* will assume that your folder directory is to be found starting from your home directory. Thus, if your home directory is **/usr/person** the above example told *Mail* to find your folder directory in **/usr/person/letters**.

Anywhere a file name is expected, you can use a folder name, preceded with '+.' For example, to put a message into a folder with the **save** command, you can use:

```
save +classwork
```

to save the current message in the *classwork* folder. If the *classwork* folder does not yet exist, it will be created. Note that messages which are saved with the **save** command are automatically removed from your system mailbox.

In order to make a copy of a message in a folder without causing that message to be removed from your system mailbox, use the **copy** command, which is identical in all other respects to the **save** command. For example,

```
copy +classwork
```

copies the current message into the *classwork* folder and leaves a copy in your system mailbox.

The **folder** command can be used to direct *Mail* to the contents of a different folder. For example,

```
folder +classwork
```

directs *Mail* to read the contents of the *classwork* folder. All of the commands that you can use on your system mailbox are also applicable to folders, including **type**, **delete**, and **reply**. To inquire which folder you are currently editing, use simply:

```
folder
```

To list your current set of folders, use the **folders** command.

To start *Mail* reading one of your folders, you can use the **−f** option described in section 2. For example:

```
% Mail −f +classwork
```

will cause *Mail* to read your *classwork* folder without looking at your system mailbox.

4. More about sending mail

4.1. Tilde escapes

While typing in a message to be sent to others, it is often useful to be able to invoke the text editor on the partial message, print the message, execute a shell command, or do some other auxiliary function. *Mail* provides these capabilities through *tilde escapes*, which consist of a tilde (~) at the beginning of a line, followed by a single character which indicates the function to be performed. For example, to print the text of the message so far, use:

```
~p
```

which will print a line of dashes, the recipients of your message, and the text of the message so far. Since *Mail* requires two consecutive RUBOUT's to abort a letter, you can use a single RUBOUT to abort the output of ~p or any other ~ escape without killing your letter.

If you are dissatisfied with the message as it stands, you can invoke the text editor on it using the escape

```
~e
```

which causes the message to be copied into a temporary file and an instance of the editor to be spawned. After modifying the message to your satisfaction, write it out and quit the editor. *Mail* will respond by typing

```
(continue)
```

after which you may continue typing text which will be appended to your message, or type <control-d> to end the message. A standard text editor is provided by *Mail*. You can override this default by setting the valued option "EDITOR" to something else. For example, you might prefer:

```
set EDITOR=/usr/ucb/ex
```

Many systems offer a screen editor as an alternative to the standard text editor, such as the *vi* editor from UC Berkeley. To use the screen, or *visual* editor, on your current message, you can use the escape,

```
~v
```

~v works like ~e, except that the screen editor is invoked instead. A default screen editor is defined by *Mail*. If it does not suit you, you can set the valued option "VISUAL" to the path name of a different editor.

It is often useful to be able to include the contents of some file in your message; the escape

```
~r filename
```

is provided for this purpose, and causes the named file to be appended to your current message. *Mail* complains if the file doesn't exist or can't be read. If the read is successful, the number of lines and characters appended to your message is printed, after which you may continue appending text. The filename may contain shell metacharacters like * and ? which are expanded according to the conventions of your shell.

As a special case of ~r, the escape

```
~d
```

reads in the file "dead.letter" in your home directory. This is often useful since *Mail* copies the text of your message there when you abort a message with RUBOUT.

To save the current text of your message on a file you may use the

```
~w filename
```

escape. *Mail* will print out the number of lines and characters written to the file, after which you may continue appending text to your message. Shell metacharacters may be used in the filename, as in ~r and are expanded with the conventions of your shell.

If you are sending mail from within *Mail's* command mode you can read a message sent to you into the message you are constructing with the escape:

```
~m 4
```

which will read message 4 into the current message, shifted right by one tab stop. You can name any non-deleted message, or list of messages. Messages can also be forwarded without shifting by a tab stop with ~f. This is the usual way to forward a message.

If, in the process of composing a message, you decide to add additional people to the list of message recipients, you can do so with the escape

```
~t name1 name2 ...
```

You may name as few or many additional recipients as you wish. Note that the users originally on the recipient list will still receive the message; you cannot remove someone from the recipient list with ~t.

If you wish, you can associate a subject with your message by using the escape

```
~s Arbitrary string of text
```

which replaces any previous subject with "Arbitrary string of text." The subject, if given, is sent near the top of the message prefixed with "Subject:" You can see what the message will look like by using ~p.

For political reasons, one occasionally prefers to list certain people as recipients of carbon copies of a message rather than direct recipients. The escape

```
~c name1 name2 ...
```

adds the named people to the "Cc:" list, similar to ~t. Again, you can execute ~p to see what the message will look like.

The recipients of the message together constitute the "To:" field, the subject the "Subject:" field, and the carbon copies the "Cc:" field. If you wish to edit these in ways impossible with the ~t, ~s, and ~c escapes, you can use the escape

```
~h
```

which prints "To:" followed by the current list of recipients and leaves the cursor (or printhead) at the end of the line. If you type in ordinary characters, they are appended to the end of the current list of recipients. You can also use your erase character to erase back into the list of recipients, or your kill character to erase them altogether. Thus, for example, if your erase and kill characters are the standard # and @ symbols,

```
~h
To: root kurt####bill
```

would change the initial recipients "root kurt" to "root bill." When you type a newline, *Mail* advances to the "Subject:" field, where the same rules apply. Another newline brings you to the "Cc:" field, which may be edited in the same fashion. Another newline leaves you appending text to the end of your message. You can use ~p to print the current text of the header fields and the body of the message.

To effect a temporary escape to the shell, the escape

```
~!command
```

is used, which executes *command* and returns you to mailing mode without altering the text of your message. If you wish, instead, to filter the body of your message through a shell command, then you can use

```
~|command
```

which pipes your message through the command and uses the output as the new text of your message. If the command produces no output, *Mail* assumes that something is amiss and retains the old version of your message. A frequently-used filter is the command *fmt*, designed to format outgoing mail.

To effect a temporary escape to *Mail* command mode instead, you can use the

```
~:Mail command
```

escape. This is especially useful for retyping the message you are replying to, using, for example:

```
~:t
```

It is also useful for setting options and modifying aliases.

If you wish (for some reason) to send a message that contains a line beginning with a tilde, you must double it. Thus, for example,

```
~~This line begins with a tilde.
```

sends the line

```
~This line begins with a tilde.
```

Finally, the escape

```
~?
```

prints out a brief summary of the available tilde escapes.

On some terminals (particularly ones with no lower case) tilde's are difficult to type. *Mail* allows you to change the escape character with the "escape" option. For example, I set

```
set escape=]
```

and use a right bracket instead of a tilde. If I ever need to send a line beginning with right bracket, I double it, just as for ~. Changing the escape character removes the special meaning of ~.

4.2. Network access

This section describes how to send mail to people on other machines. Recall that sending to a plain login name sends mail to that person on your machine. If your machine is directly (or sometimes, even, indirectly) connected to the Arpanet, you can send messages to people on the Arpanet using a name of the form

```
name@host
```

where *name* is the login name of the person you're trying to reach and *host* is the name of the machine where he logs in on the Arpanet.

If your recipient logs in on a machine connected to yours by UUCP (the Bell Laboratories supplied network that communicates over telephone lines), sending mail to him is a bit more complicated. You must know the list of machines through which your message must travel to arrive at his site. So, if his machine is directly connected to yours, you can send mail to him using the syntax:

```
host!name
```

where, again, *host* is the name of his machine and *name* is his login name. If your message must go through an intermediate machine first, you must use the syntax:

```
intermediate!host!name
```

and so on. It is actually a feature of UUCP that the map of all the systems in the network is not known anywhere (except where people decide to write it down for convenience). Talk to your system administrator about the machines connected to your site.

If you want to send a message to a recipient on the Berkeley network (Berknet), you use the syntax:

```
host:name
```

where *host* is his machine name and *name* is his login name. Unlike UUCP, you need not know the names of the intermediate machines.

When you use the **reply** command to respond to a letter, there is a problem of figuring out the names of the users in the "To:" and "Cc:" lists *relative to the current machine.* If the original letter was sent to you by someone on the local machine, then this problem does not exist, but if the message came from a remote machine, the problem must be dealt with. *Mail* uses a heuristic to build the correct name for each user relative to the local machine. So, when you **reply** to remote mail, the names in the "To:" and "Cc:" lists may change somewhat.

4.3. Special recipients

As described previously, you can send mail to either user names or **alias** names. It is also possible to send messages directly to files or to programs, using special conventions. If a recipient name has a '/' in it or begins with a '+', it is assumed to be the path name of a file into which to send the message. If the file already exists, the message is appended to the end of the file. If you want to name a file in your current directory (ie, one for which a '/' would not usually be needed) you can precede the name with './' So, to send mail to the file "memo" in the current directory, you can give the command:

```
% Mail ./memo
```

If the name begins with a '+,' it is expanded into the full path name of the folder name in your folder directory. This ability to send mail to files can be used for a variety of purposes, such as maintaining a journal and keeping a record of mail sent to a certain group of users. The second example can be done automatically by including the full pathname of the record file in the **alias** command for the group. Using our previous **alias** example, you might give the command:

```
alias project sam sally steve susan /usr/project/mail_record
```

Then, all mail sent to "project" would be saved on the file "/usr/project/mail_record" as well as being sent to the members of the project. This file can be examined using *Mail −f.*

It is sometimes useful to send mail directly to a program, for example one might write a project billboard program and want to access it using *Mail.* To send messages to the billboard program, one can send mail to the special name '|billboard' for example. *Mail* treats recipient names that begin with a '|' as a program to send the mail to. An **alias** can be set up to reference a '|' prefaced name if desired. *Caveats*: the shell treats '|' specially, so it must be quoted on the command line. Also, the '| program' must be presented as a single argument to mail. The safest course is to surround the entire name with double quotes. This also applies to usage in the **alias** command. For example, if we wanted to alias 'rmsgs' to 'rmsgs −s' we would need to say:

```
alias rmsgs "| rmsgs -s"
```

5. Additional features

This section describes some additional commands of use for reading your mail, setting options, and handling lists of messages.

5.1. Message lists

Several *Mail* commands accept a list of messages as an argument. Along with **type** and **delete**, described in section 2, there is the **from** command, which prints the message headers associated with the message list passed to it. The **from** command is particularly useful in conjunction with some of the message list features described below.

A *message list* consists of a list of message numbers, ranges, and names, separated by spaces or tabs. Message numbers may be either decimal numbers, which directly specify messages, or one of the special characters "↑" "." or "$" to specify the first relevant, current, or last relevant message, respectively. *Relevant* here means, for most commands "not deleted" and "deleted" for the **undelete** command.

A range of messages consists of two message numbers (of the form described in the previous paragraph) separated by a dash. Thus, to print the first four messages, use

```
type 1−4
```

and to print all the messages from the current message to the last message, use

```
type .−$
```

A *name* is a user name. The user names given in the message list are collected together and each message selected by other means is checked to make sure it was sent by one of the named users. If the message consists entirely of user names, then every message sent by one those users that is *relevant* (in the sense described earlier) is selected. Thus, to print every message sent to you by "root," do

```
type root
```

As a shorthand notation, you can specify simply "*" to get every *relevant* (same sense) message. Thus,

```
type *
```

prints all undeleted messages,

```
delete *
```

deletes all undeleted messages, and

```
undelete *
```

undeletes all deleted messages.

You can search for the presence of a word in subject lines with /. For example, to print the headers of all messages that contain the word "PASCAL," do:

```
from /pascal
```

Note that subject searching ignores upper/lower case differences.

5.2. List of commands

This section describes all the *Mail* commands available when receiving mail.

! Used to preface a command to be executed by the shell.

− The − command goes to the previous message and prints it. The − command may be given a decimal number *n* as an argument, in which case the *n*th previous message is gone to and printed.

PrintLike **print**, but also print out ignored header fields. See also **print** and **ignore**.

Reply
: Note the capital R in the name. Frame a reply to a one or more messages. The reply (or replies if you are using this on multiple messages) will be sent ONLY to the person who sent you the message (respectively, the set of people who sent the messages you are replying to). You can add people using the ~t and ~c tilde escapes. The subject in your reply is formed by prefacing the subject in the original message with "Re:" unless it already began thus. If the original message included a "reply-to" header field, the reply will go *only* to the recipient named by "reply-to." You type in your message using the same conventions available to you through the **mail** command. The **Reply** command is especially useful for replying to messages that were sent to enormous distribution groups when you really just want to send a message to the originator. Use it often.

Type Identical to the **Print** command.

alias Define a name to stand for a set of other names. This is used when you want to send messages to a certain group of people and want to avoid retyping their names. For example

    alias project john sue willie kathryn

creates an alias *project* which expands to the four people John, Sue, Willie, and Kathryn.

alternates
If you have accounts on several machines, you may find it convenient to use the /usr/lib/aliases on all the machines except one to direct your mail to a single account. The **alternates** command is used to inform *Mail* that each of these other addresses is really *you. Alternates* takes a list of user names and remembers that they are all actually you. When you **reply** to messages that were sent to one of these alternate names, *Mail* will not bother to send a copy of the message to this other address (which would simply be directed back to you by the alias mechanism). If *alternates* is given no argument, it lists the current set of alternate names. **Alternates** is usually used in the .mailrc file.

chdir The **chdir** command allows you to change your current directory. **Chdir** takes a single argument, which is taken to be the pathname of the directory to change to. If no argument is given, **chdir** changes to your home directory.

copy The **copy** command does the same thing that **save** does, except that it does not mark the messages it is used on for deletion when you quit.

delete
Deletes a list of messages. Deleted messages can be reclaimed with the **undelete** command.

dt The **dt** command deletes the current message and prints the next message. It is useful for quickly reading and disposing of mail.

edit To edit individual messages using the text editor, the **edit** command is provided. The **edit** command takes a list of messages as described under the **type** command and processes each by writing it into the file Message*x* where *x* is the message number being edited and executing the text editor on it. When you have edited the message to your satisfaction, write the message out and quit, upon which *Mail* will read the message back and remove the file. **Edit** may be abbreviated to **e**.

else Marks the end of the then-part of an **if** statement and the beginning of the part to take effect if the condition of the **if** statement is false.

endif Marks the end of an **if** statement.

exit Leave *Mail* without updating the system mailbox or the file your were reading. Thus, if you accidentally delete several messages, you can use **exit** to avoid scrambling your mailbox.

file The same as **folder**.

folders
List the names of the folders in your folder directory.

folder
The **folder** command switches to a new mail file or folder. With no arguments, it tells you which file you are currently reading. If you give it an argument, it will write out changes (such as deletions) you have made in the current file and read the new file. Some special conventions are recognized for the name:

Name	Meaning
#	Previous file read
%	Your system mailbox
%name	*Name*'s system mailbox
&	Your ~/mbox file
+folder	A file in your folder directory

from The **from** command takes a list of messages and prints out the header lines for each one; hence

```
from joe
```

is the easy way to display all the message headers from "joe."

headers
When you start up *Mail* to read your mail, it lists the message headers that you have. These headers tell you who each message is from, when they were sent, how many lines and characters each message is, and the "Subject:" header field of each message, if present. In addition, *Mail* tags the message header of each message that has been the object of the **preserve** command with a "P." Messages that have been **saved** or **written** are flagged with a "*." Finally, **deleted** messages are not printed at all. If you wish to reprint the current list of message headers, you can do so with the **headers** command. The **headers** command (and thus the initial header listing) only lists the first so many message headers. The number of headers listed depends on the speed of your terminal. This can be overridden by specifying the number of headers you want with the *window* option. *Mail* maintains a notion of the current "window" into your messages for the purposes of printing headers. Use the **z** command to move forward and back a window. You can move *Mail's* notion of the current window directly to a particular message by using, for example,

```
headers 40
```

to move *Mail's* attention to the messages around message 40. The **headers** command can be abbreviated to **h**.

help Print a brief and usually out of date help message about the commands in *Mail*. Refer to this manual instead.

hold Arrange to hold a list of messages in the system mailbox, instead of moving them to the file *mbox* in your home directory. If you set the binary option *hold*, this will happen by default.

if Commands in your ".mailrc" file can be executed conditionally depending on whether you are sending or receiving mail with the **if** command. For example, you can do:

```
if receive
        commands...
endif
```

An **else** form is also available:

```
if send
        commands...
else
```

```
        commands...
    endif
```

Note that the only allowed conditions are **receive** and **send**.

ignore
Add the list of header fields named to the *ignore list.* Header fields in the ignore list are not printed on your terminal when you print a message. This allows you to suppress printing of certain machine-generated header fields, such as *Via* which are not usually of interest. The **Type** and **Print** commands can be used to print a message in its entirety, including ignored fields. If **ignore** is executed with no arguments, it lists the current set of ignored fields.

list List the vaild *Mail* commands.

mail Send mail to one or more people. If you have the *ask* option set, *Mail* will prompt you for a subject to your message. Then you can type in your message, using tilde escapes as described in section 4 to edit, print, or modify your message. To signal your satisfaction with the message and send it, type control-d at the beginning of a line, or a . alone on a line if you set the option *dot.* To abort the message, type two interrupt characters (RUBOUT by default) in a row or use the **~q** escape.

mbox
Indicate that a list of messages be sent to *mbox* in your home directory when you quit. This is the default action for messages if you do *not* have the *hold* option set.

next The **next** command goes to the next message and types it. If given a message list, **next** goes to the first such message and types it. Thus,

```
next root
```

goes to the next message sent by "root" and types it. The **next** command can be abbreviated to simply a newline, which means that one can go to and type a message by simply giving its message number or one of the magic characters "↑" "." or "$". Thus,

prints the current message and

```
4
```

prints message 4, as described previously.

preserve
Same as **hold**. Cause a list of messages to be held in your system mailbox when you quit.

quit Leave *Mail* and update the file, folder, or system mailbox your were reading. Messages that you have examined are marked as "read" and messages that existed when you started are marked as "old." If you were editing your system mailbox and if you have set the binary option *hold,* all messages which have not been deleted, saved, or mboxed will be retained in your system mailbox. If you were editing your system mailbox and you did *not* have *hold* set, all messages which have not been deleted, saved, or preserved will be moved to the file *mbox* in your home directory.

reply Frame a reply to a single message. The reply will be sent to the person who sent you the message to which you are replying, plus all the people who received the original message, except you. You can add people using the ~t and ~c tilde escapes. The subject in your reply is formed by prefacing the subject in the original message with "Re:" unless it already began thus. If the original message included a "reply-to" header field, the reply will go *only* to the recipient named by "reply-to." You type in your message using the same conventions available to you through the **mail** command.

save It is often useful to be able to save messages on related topics in a file. The **save** command gives you ability to do this. The **save** command takes as argument a lit of message numbers, followed by the name of the file on which to save the messages. The messages are appended to the named file, thus allowing one to keep several messages in the file, stored in the order they were put there. The **save** command can be abbreviated to **s**. An example of the **save** command relative to our running example is:

```
s 1 2 tuitionmail
```

Saved messages are not automatically saved in *mbox* at quit time, nor are they selected by the **next** command described above, unless explicitly specified.

set Set an option or give an option a value. Used to customize *Mail*. Section 5.3 contains a list of the options. Options can be *binary*, in which case they are *on* or *off*, or *valued*. To set a binary option *option on*, do

```
set option
```

To give the valued option *option* the value *value*, do

```
set option=value
```

Several options can be specified in a single **set** command.

shell The **shell** command allows you to escape to the shell. **Shell** invokes an interactive shell and allows you to type commands to it. When you leave the shell, you will return to *Mail*. The shell used is a default assumed by *Mail*; you can override this default by setting the valued option "SHELL," eg:

```
set SHELL=/bin/csh
```

source
The **source** command reads *Mail* commands from a file. It is useful when you are trying to fix your ".mailrc" file and you need to re-read it.

top The **top** command takes a message list and prints the first five lines of each addressed message. It may be abbreviated to **to**. If you wish, you can change the number of lines that **top** prints out by setting the valued option "toplines." On a CRT terminal,

```
set toplines=10
```

might be preferred.

type Print a list of messages on your terminal. If you have set the option *crt* to a number and the total number of lines in the messages you are printing exceed that specified by *crt*, the messages will be printed by a terminal paging program such as *more*.

undelete
The **undelete** command causes a message that had been deleted previously to regain its initial status. Only messages that have been deleted may be undeleted. This command may be abbreviated to **u**.

unset
Reverse the action of setting a binary or valued option.

visual
It is often useful to be able to invoke one of two editors, based on the type of terminal one is using. To invoke a display oriented editor, you can use the **visual** command. The operation of the **visual** command is otherwise identical to that of the **edit** command.

Both the **edit** and **visual** commands assume some default text editors. These default editors can be overridden by the valued options "EDITOR" and "VISUAL" for the standard and screen editors. You might want to do:

```
set EDITOR=/usr/ucb/ex VISUAL=/usr/ucb/vi
```

write The **save** command always writes the entire message, including the headers, into the file. If you want to write just the message itself, you can use the **write** command. The **write** command has the same syntax as the **save** command, and can be abbreviated to simply **w**. Thus, we could write the second message by doing:

```
w 2 file.c
```

As suggested by this example, the **write** command is useful for such tasks as sending and receiving source program text over the message system.

z *Mail* presents message headers in windowfuls as described under the **headers** command. You can move *Mail's* attention forward to the next window by giving the

```
z+
```

command. Analogously, you can move to the previous window with:

```
z−
```

5.3. Custom options

Throughout this manual, we have seen examples of binary and valued options. This section describes each of the options in alphabetical order, including some that you have not seen yet. To avoid confusion, please note that the options are either all lower case letters or all upper case letters. When I start a sentence such as: "Ask" causes *Mail* to prompt you for a subject header, I am only capitalizing "ask" as a courtesy to English.

EDITOR
The valued option "EDITOR" defines the pathname of the text editor to be used in the **edit** command and ˜e. If not defined, a standard editor is used.

SHELL
The valued option "SHELL" gives the path name of your shell. This shell is used for the ! command and ˜! escape. In addition, this shell expands file names with shell metacharacters like * and ? in them.

VISUAL
The valued option "VISUAL" defines the pathname of your screen editor for use in the **visual** command and ˜v escape. A standard screen editor is used if you do not define one.

append
The "append" option is binary and causes messages saved in *mbox* to be appended to the end rather than prepended. Normally, *Mail*will *mbox* in the same order that the system puts messages in your system mailbox. By setting "append," you are requesting that *mbox* be appended to regardless. It is in any event quicker to append.

ask "Ask" is a binary option which causes *Mail* to prompt you for the subject of each message you send. If you respond with simply a newline, no subject field will be sent.

askcc
"Askcc" is a binary option which causes you to be prompted for additional carbon copy recipients at the end of each message. Responding with a newline shows your satisfaction with the current list.

autoprint
"Autoprint" is a binary option which causes the **delete** command to behave like **dp** — thus, after deleting a message, the next one will be typed automatically. This is useful to quickly scanning and deleting messages in your mailbox.

debug
The binary option "debug" causes debugging information to be displayed. Use of this option is the same as useing the

−d command line flag.

dot "Dot" is a binary option which, if set, causes *Mail* to interpret a period alone on a line as the terminator of a message you are sending.

escape
To allow you to change the escape character used when sending mail, you can set the valued option "escape." Only the first character of the "escape" option is used, and it

must be doubled if it is to appear as the first character of a line of your message. If you change your escape character, then ˜ loses all its special meaning, and need no longer be doubled at the beginning of a line.

folder
The name of the directory to use for storing folders of messages. If this name begins with a '/' *Mail* considers it to be an absolute pathname; otherwise, the folder directory is found relative to your home directory.

hold The binary option "hold" causes messages that have been read but not manually dealt with to be held in the system mailbox. This prevents such messages from being automatically swept into your mbox.

ignore
The binary option "ignore" causes RUBOUT characters from your terminal to be ignored and echoed as @'s while you are sending mail. RUBOUT characters retain their original meaning in *Mail* command mode. Setting the "ignore" option is equivalent to supplying the **−i** flag on the command line as described in section 6.

ignoreeof
An option related to "dot" is "ignoreeof" which makes *Mail* refuse to accept a control−d as the end of a message. "Ignoreeof" also applies to *Mail* command mode.

keep The "keep" option causes *Mail* to truncate your system mailbox instead of deleting it when it is empty. This is useful if you elect to protect your mailbox, which you would do with the shell command:

```
chmod 600 /usr/spool/mail/yourname
```

where *yourname* is your login name. If you do not do this, anyone can probably read your mail, although people usually don't.

keepsave
When you **save** a message, *Mail* usually discards it when you **quit**. To retain all saved messages, set the "keepsave" option.

metoo
When sending mail to an alias, *Mail* makes sure that if you are included in the alias, that mail will not be sent to you. This is useful if a single alias is being used by all members of the group. If however, you wish to receive a copy of all the messages you send to the alias, you can set the binary option "metoo."

noheader
The binary option "noheader" suppresses the printing of the version and headers when *Mail* is first invoked. Setting this option is the same as using **−N** on the command line.

nosave
Normally, when you abort a message with two RUBOUTs, *Mail* copies the partial letter to the file "dead.letter" in your home directory. Setting the binary option "nosave" prevents this.

quiet The binary option "quiet" suppresses the printing of the version when *Mail* is first invoked, as well as printing the for example "Message 4:" from the **type** command.

record
If you love to keep records, then the valued option "record" can be set to the name of a file to save your outgoing mail. Each new message you send is appended to the end of the file.

screen
When *Mail* initially prints the message headers, it determines the number to print by looking at the speed of your terminal. The faster your terminal, the more it prints. The valued option "screen" overrides this calculation and specifies how many message headers you want printed. This number is also used for scrolling with the **z** command.

sendmail
: To alternate delivery system, set the "sendmail" option to the full pathname of the program to use. Note: this is not for everyone! Most people should use the default delivery system.

toplines
: The valued option "toplines" defines the number of lines that the "top" command will print out instead of the default five lines.

verbose
: The binary option "verbose" causes *Mail* to invoke sendmail with the **−v** flag, which causes it to go into versbose mode and announce expansion of aliases, etc. Setting the "verbose" option is equivalent to invoking *Mail* with the **−v** flag as described in section 6.

6. Command line options

This section describes command line options for *Mail* and what they are used for.

−N Suppress the initial printing of headers.

−d Turn on debugging information. Not of general interest.

−f file
Show the messages in *file* instead of your system mailbox. If *file* is omitted, *Mail* reads *mbox* in your home directory.

−i Ignore tty interrupt signals. Useful on noisy phone lines, which generate spurious RUBOUT or DELETE characters. It's usually more effective to change your interrupt character to control−c, for which see the *stty* shell command.

−n Inhibit reading of /usr/lib/Mail.rc. Not generally useful, since /usr/lib/Mail.rc is usually empty.

−s string
Used for sending mail. *String* is used as the subject of the message being composed. If *string* contains blanks, you must surround it with quote marks.

−u name
Read *names's* mail instead of your own. Unwitting others often neglect to protect their mailboxes, but discretion is advised. Essentially, **−u user** is a shorthand way of doing **−f /usr/spool/user**.

−v Use the **−v** flag when invoking sendmail. This feature may also be enabled by setting the the option "verbose".

The following command line flags are also recognized, but are intended for use by programs invoking *Mail* and not for people.

−T file
Arrange to print on *file* the contents of the *article-id* fields of all messages that were either read or deleted. **−T** is for the *readnews* program and should NOT be used for reading your mail.

−h number
Pass on hop count information. *Mail* will take the number, increment it, and pass it with **−h** to the mail delivery system. **−h** only has effect when sending mail and is used for network mail forwarding.

−r name
Used for network mail forwarding: interpret *name* as the sender of the message. The *name* and **−r** are simply sent along to the mail delivery system. Also, *Mail* will wait for the message to be sent and return the exit status. Also restricts formatting of message.

Note that **−h** and **−r**, which are for network mail forwarding, are not used in practice since mail forwarding is now handled separately. They may disappear soon.

7. Format of messages

This section describes the format of messages. Messages begin with a *from* line, which consists of the word "From" followed by a user name, followed by anything, followed by a date in the format returned by the *ctime* library routine described in section 3 of the Unix Programmer's Manual. A possible *ctime* format date is:

Tue Dec 1 10:58:23 1981

The *ctime* date may be optionally followed by a single space and a time zone indication, which should be three capital letters, such as PDT.

Following the *from* line are zero or more *header field* lines. Each header field line is of the form:

name: information

Name can be anything, but only certain header fields are recognized as having any meaning. The recognized header fields are: *article-id*, *bcc*, *cc*, *from*, *reply-to*, *sender*, *subject*, and *to*. Other header fields are also significant to other systems; see, for example, the current Arpanet message standard for much more on this topic. A header field can be continued onto following lines by making the first character on the following line a space or tab character.

If any headers are present, they must be followed by a blank line. The part that follows is called the *body* of the message, and must be ASCII text, not containing null characters. Each line in the message body must be terminated with an ASCII newline character and no line may be longer than 512 characters. If binary data must be passed through the mail system, it is suggested that this data be encoded in a system which encodes six bits into a printable character. For example, one could use the upper and lower case letters, the digits, and the characters comma and period to make up the 64 characters. Then, one can send a 16-bit binary number as three characters. These characters should be packed into lines, preferably lines about 70 characters long as long lines are transmitted more efficiently.

The message delivery system always adds a blank line to the end of each message. This blank line must not be deleted.

The UUCP message delivery system sometimes adds a blank line to the end of a message each time it is forwarded through a machine.

It should be noted that some network transport protocols enforce limits to the lengths of messages.

8. Glossary

This section contains the definitions of a few phrases peculiar to *Mail.*

alias An alternative name for a person or list of people.

flag An option, given on the command line of *Mail,* prefaced with a −. For example, **−f** is a flag.

header field
At the beginning of a message, a line which contains information that is part of the structure of the message. Popular header fields include *to*, *cc*, and *subject.*

mail A collection of messages. Often used in the phrase, "Have you read your mail?"

mailbox
The place where your mail is stored, typically in the directory /usr/spool/mail.

message
A single letter from someone, initially stored in your *mailbox.*

message list
A string used in *Mail* command mode to describe a sequence of messages.

option
A piece of special purpose information used to tailor *Mail* to your taste. Options are specified with the set command.

9. Summary of commands, options, and escapes

This section gives a quick summary of the *Mail* commands, binary and valued options, and tilde escapes.

The following table describes the commands:

Command	*Description*
!	Single command escape to shell
-	Back up to previous message
Print	Type message with ignored fields
Reply	Reply to author of message only
Type	Type message with ignored fields
alias	Define an alias as a set of user names
alternates	List other names you are known by
chdir	Change working directory, home by default
copy	Copy a message to a file or folder
delete	Delete a list of messages
dt	Delete current message, type next message
endif	End of conditional statement; see **if**
edit	Edit a list of messages
else	Start of else part of conditional; see **if**
exit	Leave mail without changing anything
file	Interrogate/change current mail file
folder	Same as **file**
folders	List the folders in your folder directory
from	List headers of a list of messages
headers	List current window of messages
help	Print brief summary of *Mail* commands
hold	Same as **preserve**
if	Conditional execution of *Mail* commands
ignore	Set/examine list of ignored header fields
list	List valid *Mail* commands
local	List other names for the local host
mail	Send mail to specified names
mbox	Arrange to save a list of messages in *mbox*
next	Go to next message and type it
preserve	Arrange to leave list of messages in system mailbox
quit	Leave *Mail*; update system mailbox, *mbox* as appropriate
reply	Compose a reply to a message
save	Append messages, headers included, on a file
set	Set binary or valued options
shell	Invoke an interactive shell
top	Print first so many (5 by default) lines of list of messages
type	Print messages
undelete	Undelete list of messages
unset	Undo the operation of a set
visual	Invoke visual editor on a list of messages
write	Append messages to a file, don't include headers
z	Scroll to next/previous screenful of headers

The following table describes the options. Each option is shown as being either a binary or valued option.

Option	*Type*	*Description*
EDITOR	*valued*	Pathname of editor for ~e and **edit**
SHELL	*valued*	Pathname of shell for **shell**, ~! and !
VISUAL	*valued*	Pathname of screen editor for ~v, **visual**
append	*binary*	Always append messages to end of *mbox*
ask	*binary*	Prompt user for Subject: field when sending
askcc	*binary*	Prompt user for additional Cc's at end of message
autoprint	*binary*	Print next message after **delete**
crt	*valued*	Minimum number of lines before using *more*
debug	*binary*	Print out debugging information
dot	*binary*	Accept . alone on line to terminate message input
escape	*valued*	Escape character to be used instead of ~
folder	*valued*	Directory to store folders in
hold	*binary*	Hold messages in system mailbox by default
ignore	*binary*	Ignore RUBOUT while sending mail
ignoreeof	*binary*	Don't terminate letters/command input with ↑D
keep	*binary*	Don't unlink system mailbox when empty
keepsave	*binary*	Don't delete **saved** messages by default
metoo	*binary*	Include sending user in aliases
noheader	*binary*	Suppress initial printing of version and headers
nosave	*binary*	Don't save partial letter in *dead.letter*
quiet	*binary*	Suppress printing of *Mail* version and message numbers
record	*valued*	File to save all outgoing mail in
screen	*valued*	Size of window of message headers for z, etc.
sendmail	*valued*	Choose alternate mail delivery system
toplines	*valued*	Number of lines to print in **top**
verbose	*binary*	Invoke sendmail with the **−v** flag

The following table summarizes the tilde escapes available while sending mail.

Escape	*Arguments*	*Description*
~!	*command*	Execute shell command
~c	*name ...*	Add names to Cc: field
~d		Read *dead.letter* into message
~e		Invoke text editor on partial message
~f	*messages*	Read named messages
~h		Edit the header fields
~m	*messages*	Read named messages, right shift by tab
~p		Print message entered so far
~q		Abort entry of letter; like RUBOUT
~r	*filename*	Read file into message
~s	*string*	Set Subject: field to *string*
~t	*name ...*	Add names to To: field
~v		Invoke screen editor on message
~w	*filename*	Write message on file
~|	*command*	Pipe message through *command*
~~	*string*	Quote a ~ in front of *string*

The following table shows the command line flags that *Mail* accepts:

Flag	Description
−N	Suppress the initial printing of headers
−T *file*	Article-id's of read/deleted messages to *file*
−d	Turn on debugging
−f *file*	Show messages in *file* or *~/mbox*
−h *number*	Pass on hop count for mail forwarding
−i	Ignore tty interrupt signals
−n	Inhibit reading of /usr/lib/Mail.rc
−r *name*	Pass on *name* for mail forwarding
−s *string*	Use *string* as subject in outgoing mail
−u *name*	Read *name's* mail instead of your own
−v	Invoke sendmail with the **−v** flag

Notes: **−T**, **−d**, **−h**, and **−r** are not for human use.

10. Conclusion

Mail is an attempt to provide a simple user interface to a variety of underlying message systems. Thanks are due to the many users who contributed ideas and testing to *Mail*.

SENDMAIL — An Internetwork Mail Router

Eric Allman†

Britton-Lee, Inc.
1919 Addison Street, Suite 105.
Berkeley, California 94704.

ABSTRACT

Routing mail through a heterogenous internet presents many new problems. Among the worst of these is that of address mapping. Historically, this has been handled on an *ad hoc* basis. However, this approach has become unmanageable as internets grow.

Sendmail acts a unified "post office" to which all mail can be submitted. Address interpretation is controlled by a production system, which can parse both domain-based addressing and old-style *ad hoc* addresses. The production system is powerful enough to rewrite addresses in the message header to conform to the standards of a number of common target networks, including old (NCP/RFC733) Arpanet, new (TCP/RFC822) Arpanet, UUCP, and Phonenet. Sendmail also implements an SMTP server, message queueing, and aliasing.

Sendmail implements a general internetwork mail routing facility, featuring aliasing and forwarding, automatic routing to network gateways, and flexible configuration.

In a simple network, each node has an address, and resources can be identified with a host-resource pair; in particular, the mail system can refer to users using a host-username pair. Host names and numbers have to be administered by a central authority, but usernames can be assigned locally to each host.

In an internet, multiple networks with different characterstics and managements must communicate. In particular, the syntax and semantics of resource identification change. Certain special cases can be handled trivially by *ad hoc* techniques, such as providing network names that appear local to hosts on other networks, as with the Ethernet at Xerox PARC. However, the general case is extremely complex. For example, some networks require point-to-point routing, which simplifies the database update problem since only adjacent hosts must be entered into the system tables, while others use end-to-end addressing. Some networks use a left-associative syntax and others use a right-associative syntax, causing ambiguity in mixed addresses.

Internet standards seek to eliminate these problems. Initially, these proposed expanding the address pairs to address triples, consisting of {network, host, resource} triples. Network numbers must be universally agreed upon, and hosts can be assigned locally on each network. The user-level presentation was quickly expanded to address domains, comprised of a local resource identification and a hierarchical domain specification with a common static root. The domain technique separates the issue of physical versus logical addressing. For example, an address of the form "eric@a.cc.berkeley.arpa" describes only the logical organization of the address space.

†A considerable part of this work was done while under the employ of the INGRES Project at the University of California at Berkeley.

Sendmail is intended to help bridge the gap between the totally *ad hoc* world of networks that know nothing of each other and the clean, tightly-coupled world of unique network numbers. It can accept old arbitrary address syntaxes, resolving ambiguities using heuristics specified by the system administrator, as well as domain-based addressing. It helps guide the conversion of message formats between disparate networks. In short, *sendmail* is designed to assist a graceful transition to consistent internetwork addressing schemes.

Section 1 discusses the design goals for *sendmail*. Section 2 gives an overview of the basic functions of the system. In section 3, details of usage are discussed. Section 4 compares *sendmail* to other internet mail routers, and an evaluation of *sendmail* is given in section 5, including future plans.

1. DESIGN GOALS

Design goals for *sendmail* include:

(1) Compatibility with the existing mail programs, including Bell version 6 mail, Bell version 7 mail [UNIX83], Berkeley *Mail* [Shoens79], BerkNet mail [Schmidt79], and hopefully UUCP mail [Nowitz78a, Nowitz78b]. ARPANET mail [Crocker77a, Postel77] was also required.

(2) Reliability, in the sense of guaranteeing that every message is correctly delivered or at least brought to the attention of a human for correct disposal; no message should ever be completely lost. This goal was considered essential because of the emphasis on mail in our environment. It has turned out to be one of the hardest goals to satisfy, especially in the face of the many anomalous message formats produced by various ARPANET sites. For example, certain sites generate improperly formated addresses, occasionally causing error-message loops. Some hosts use blanks in names, causing problems with UNIX mail programs that assume that an address is one word. The semantics of some fields are interpreted slightly differently by different sites. In summary, the obscure features of the ARPANET mail protocol really *are* used and are difficult to support, but must be supported.

(3) Existing software to do actual delivery should be used whenever possible. This goal derives as much from political and practical considerations as technical.

(4) Easy expansion to fairly complex environments, including multiple connections to a single network type (such as with multiple UUCP or Ether nets [Metcalfe76]). This goal requires consideration of the contents of an address as well as its syntax in order to determine which gateway to use. For example, the ARPANET is bringing up the TCP protocol to replace the old NCP protocol. No host at Berkeley runs both TCP and NCP, so it is necessary to look at the ARPANET host name to determine whether to route mail to an NCP gateway or a TCP gateway.

(5) Configuration should not be compiled into the code. A single compiled program should be able to run as is at any site (barring such basic changes as the CPU type or the operating system). We have found this seemingly unimportant goal to be critical in real life. Besides the simple problems that occur when any program gets recompiled in a different environment, many sites like to "fiddle" with anything that they will be recompiling anyway.

(6) *Sendmail* must be able to let various groups maintain their own mailing lists, and let individuals specify their own forwarding, without modifying the system alias file.

(7) Each user should be able to specify which mailer to execute to process mail being delivered for him. This feature allows users who are using specialized mailers that use a different format to build their environment without changing the system, and facilitates specialized functions (such as returning an "I am on vacation" message).

(8) Network traffic should be minimized by batching addresses to a single host where possible, without assistance from the user.

These goals motivated the architecture illustrated in figure 1. The user interacts with a mail generating and sending program. When the mail is created, the generator calls *sendmail*, which routes the message to the correct mailer(s). Since some of the senders may be network servers and some of the mailers may be network clients, *sendmail* may be used as an internet mail gateway.

2. OVERVIEW

2.1. System Organization

Sendmail neither interfaces with the user nor does actual mail delivery. Rather, it collects a message generated by a user interface program (UIP) such as Berkeley *Mail*, MS [Crocker77b], or MH [Borden79], edits the message as required by the destination network, and calls appropriate mailers to do mail delivery or queueing for network transmission[1]. This discipline allows the insertion of new mailers at minimum cost. In this sense *sendmail* resembles the Message Processing Module (MPM) of [Postel79b].

2.2. Interfaces to the Outside World

There are three ways *sendmail* can communicate with the outside world, both in receiving and in sending mail. These are using the conventional UNIX argument vector/return status, speaking SMTP over a pair of UNIX pipes, and speaking SMTP over an interprocess(or) channel.

2.2.1. Argument vector/exit status

This technique is the standard UNIX method for communicating with the process. A list of recipients is sent in the argument vector, and the message body is sent

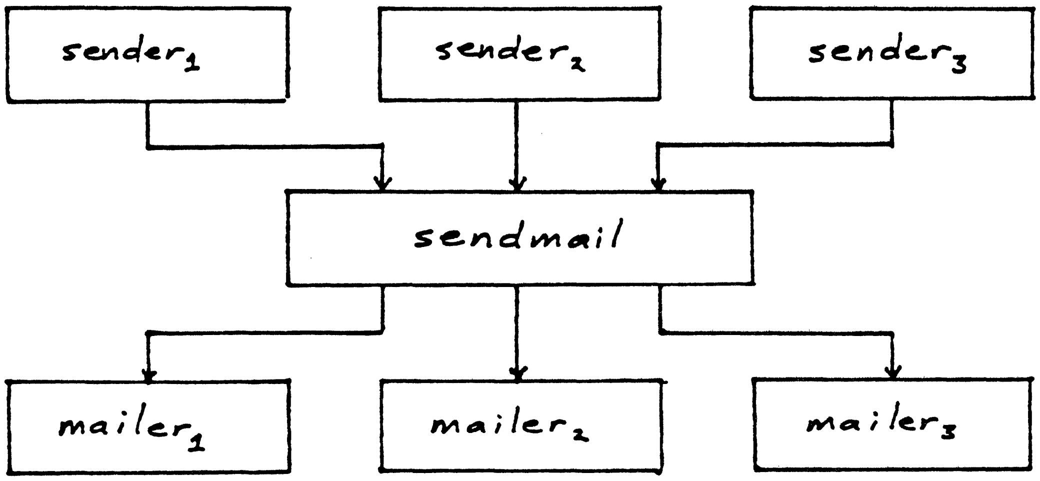

Figure 1 — Sendmail System Structure.

[1]except when mailing to a file, when *sendmail* does the delivery directly.

on the standard input. Anything that the mailer prints is simply collected and sent back to the sender if there were any problems. The exit status from the mailer is collected after the message is sent, and a diagnostic is printed if appropriate.

2.2.2. SMTP over pipes

The SMTP protocol [Postel82] can be used to run an interactive lock-step interface with the mailer. A subprocess is still created, but no recipient addresses are passed to the mailer via the argument list. Instead, they are passed one at a time in commands sent to the processes standard input. Anything appearing on the standard output must be a reply code in a special format.

2.2.3. SMTP over an IPC connection

This technique is similar to the previous technique, except that it uses a 4.2BSD IPC channel [UNIX83]. This method is exceptionally flexible in that the mailer need not reside on the same machine. It is normally used to connect to a sendmail process on another machine.

2.3. Operational Description

When a sender wants to send a message, it issues a request to *sendmail* using one of the three methods described above. *Sendmail* operates in two distinct phases. In the first phase, it collects and stores the message. In the second phase, message delivery occurs. If there were errors during processing during the second phase, *sendmail* creates and returns a new message describing the error and/or returns an status code telling what went wrong.

2.3.1. Argument processing and address parsing

If *sendmail* is called using one of the two subprocess techniques, the arguments are first scanned and option specifications are processed. Recipient addresses are then collected, either from the command line or from the SMTP RCPT command, and a list of recipients is created. Aliases are expanded at this step, including mailing lists. As much validation as possible of the addresses is done at this step: syntax is checked, and local addresses are verified, but detailed checking of host names and addresses is deferred until delivery. Forwarding is also performed as the local addresses are verified.

Sendmail appends each address to the recipient list after parsing. When a name is aliased or forwarded, the old name is retained in the list, and a flag is set that tells the delivery phase to ignore this recipient. This list is kept free from duplicates, preventing alias loops and duplicate messages deliverd to the same recipient, as might occur if a person is in two groups.

2.3.2. Message collection

Sendmail then collects the message. The message should have a header at the beginning. No formatting requirements are imposed on the message except that they must be lines of text (i.e., binary data is not allowed). The header is parsed and stored in memory, and the body of the message is saved in a temporary file.

To simplify the program interface, the message is collected even if no addresses were valid. The message will be returned with an error.

2.3.3. Message delivery

For each unique mailer and host in the recipient list, *sendmail* calls the appropriate mailer. Each mailer invocation sends to all users receiving the message on one host. Mailers that only accept one recipient at a time are handled properly.

The message is sent to the mailer using one of the same three interfaces used to submit a message to sendmail. Each copy of the message is prepended by a customized header. The mailer status code is caught and checked, and a suitable error message given as appropriate. The exit code must conform to a system standard or a generic message ("Service unavailable") is given.

2.3.4. Queueing for retransmission

If the mailer returned an status that indicated that it might be able to handle the mail later, *sendmail* will queue the mail and try again later.

2.3.5. Return to sender

If errors occur during processing, *sendmail* returns the message to the sender for retransmission. The letter can be mailed back or written in the file "dead.letter" in the sender's home directory[2].

2.4. Message Header Editing

Certain editing of the message header occurs automatically. Header lines can be inserted under control of the configuration file. Some lines can be merged; for example, a "From:" line and a "Full-name:" line can be merged under certain circumstances.

2.5. Configuration File

Almost all configuration information is read at runtime from an ASCII file, encoding macro definitions (defining the value of macros used internally), header declarations (telling sendmail the format of header lines that it will process specially, i.e., lines that it will add or reformat), mailer definitions (giving information such as the location and characteristics of each mailer), and address rewriting rules (a limited production system to rewrite addresses which is used to parse and rewrite the addresses).

To improve performance when reading the configuration file, a memory image can be provided. This provides a "compiled" form of the configuration file.

3. USAGE AND IMPLEMENTATION

3.1. Arguments

Arguments may be flags and addresses. Flags set various processing options. Following flag arguments, address arguments may be given, unless we are running in SMTP mode. Addresses follow the syntax in RFC822 [Crocker82] for ARPANET address formats. In brief, the format is:

(1) Anything in parentheses is thrown away (as a comment).

(2) Anything in angle brackets ("< >") is preferred over anything else. This rule implements the ARPANET standard that addresses of the form

    user name <machine-address>

will send to the electronic "machine-address" rather than the human "user name."

(3) Double quotes (") quote phrases; backslashes quote characters. Backslashes are more powerful in that they will cause otherwise equivalent phrases to compare differently — for example, *user* and "*user*" are equivalent, but \user is different from either of them.

[2]Obviously, if the site giving the error is not the originating site, the only reasonable option is to mail back to the sender. Also, there are many more error disposition options, but they only effect the error message — the "return to sender" function is always handled in one of these two ways.

Parentheses, angle brackets, and double quotes must be properly balanced and nested. The rewriting rules control remaining parsing[3].

3.2. Mail to Files and Programs

Files and programs are legitimate message recipients. Files provide archival storage of messages, useful for project administration and history. Programs are useful as recipients in a variety of situations, for example, to maintain a public repository of systems messages (such as the Berkeley *msgs* program, or the MARS system [Sattley78]).

Any address passing through the initial parsing algorithm as a local address (i.e, not appearing to be a valid address for another mailer) is scanned for two special cases. If prefixed by a vertical bar ("|") the rest of the address is processed as a shell command. If the user name begins with a slash mark ("/") the name is used as a file name, instead of a login name.

Files that have setuid or setgid bits set but no execute bits set have those bits honored if *sendmail* is running as root.

3.3. Aliasing, Forwarding, Inclusion

Sendmail reroutes mail three ways. Aliasing applies system wide. Forwarding allows each user to reroute incoming mail destined for that account. Inclusion directs *sendmail* to read a file for a list of addresses, and is normally used in conjunction with aliasing.

3.3.1. Aliasing

Aliasing maps names to address lists using a system-wide file. This file is indexed to speed access. Only names that parse as local are allowed as aliases; this guarantees a unique key (since there are no nicknames for the local host).

3.3.2. Forwarding

After aliasing, recipients that are local and valid are checked for the existence of a ".forward" file in their home directory. If it exists, the message is *not* sent to that user, but rather to the list of users in that file. Often this list will contain only one address, and the feature will be used for network mail forwarding.

Forwarding also permits a user to specify a private incoming mailer. For example, forwarding to:

    "|/usr/local/newmail myname"

will use a different incoming mailer.

3.3.3. Inclusion

Inclusion is specified in RFC 733 [Crocker77a] syntax:

    :Include: pathname

An address of this form reads the file specified by *pathname* and sends to all users listed in that file.

The intent is *not* to support direct use of this feature, but rather to use this as a subset of aliasing. For example, an alias of the form:

    project: :include:/usr/project/userlist

is a method of letting a project maintain a mailing list without interaction with the system administration, even if the alias file is protected.

[3]Disclaimer: Some special processing is done after rewriting local names; see below.

It is not necessary to rebuild the index on the alias database when a :include: list is changed.

3.4. Message Collection

Once all recipient addresses are parsed and verified, the message is collected. The message comes in two parts: a message header and a message body, separated by a blank line.

The header is formatted as a series of lines of the form

field-name: field-value

Field-value can be split across lines by starting the following lines with a space or a tab. Some header fields have special internal meaning, and have appropriate special processing. Other headers are simply passed through. Some header fields may be added automatically, such as time stamps.

The body is a series of text lines. It is completely uninterpreted and untouched, except that lines beginning with a dot have the dot doubled when transmitted over an SMTP channel. This extra dot is stripped by the receiver.

3.5. Message Delivery

The send queue is ordered by receiving host before transmission to implement message batching. Each address is marked as it is sent so rescanning the list is safe. An argument list is built as the scan proceeds. Mail to files is detected during the scan of the send list. The interface to the mailer is performed using one of the techniques described in section 2.2.

After a connection is established, *sendmail* makes the per-mailer changes to the header and sends the result to the mailer. If any mail is rejected by the mailer, a flag is set to invoke the return-to-sender function after all delivery completes.

3.6. Queued Messages

If the mailer returns a "temporary failure" exit status, the message is queued. A control file is used to describe the recipients to be sent to and various other parameters. This control file is formatted as a series of lines, each describing a sender, a recipient, the time of submission, or some other salient parameter of the message. The header of the message is stored in the control file, so that the associated data file in the queue is just the temporary file that was originally collected.

3.7. Configuration

Configuration is controlled primarily by a configuration file read at startup. *Sendmail* should not need to be recomplied except

(1) To change operating systems (V6, V7/32V, 4BSD).

(2) To remove or insert the DBM (UNIX database) library.

(3) To change ARPANET reply codes.

(4) To add headers fields requiring special processing.

Adding mailers or changing parsing (i.e., rewriting) or routing information does not require recompilation.

If the mail is being sent by a local user, and the file ".mailcf" exists in the sender's home directory, that file is read as a configuration file after the system configuration file. The primary use of this feature is to add header lines.

The configuration file encodes macro definitions, header definitions, mailer definitions, rewriting rules, and options.

3.7.1. Macros

Macros can be used in three ways. Certain macros transmit unstructured textual information into the mail system, such as the name *sendmail* will use to identify itself in error messages. Other macros transmit information from *sendmail* to the configuration file for use in creating other fields (such as argument vectors to mailers); e.g., the name of the sender, and the host and user of the recipient. Other macros are unused internally, and can be used as shorthand in the configuration file.

3.7.2. Header declarations

Header declarations inform *sendmail* of the format of known header lines. Knowledge of a few header lines is built into *sendmail*, such as the "From:" and "Date:" lines.

Most configured headers will be automatically inserted in the outgoing message if they don't exist in the incoming message. Certain headers are suppressed by some mailers.

3.7.3. Mailer declarations

Mailer declarations tell *sendmail* of the various mailers available to it. The definition specifies the internal name of the mailer, the pathname of the program to call, some flags associated with the mailer, and an argument vector to be used on the call; this vector is macro-expanded before use.

3.7.4. Address rewriting rules

The heart of address parsing in *sendmail* is a set of rewriting rules. These are an ordered list of pattern-replacement rules, (somewhat like a production system, except that order is critical), which are applied to each address. The address is rewritten textually until it is either rewritten into a special canonical form (i.e., a (mailer, host, user) 3-tuple, such as {arpanet, usc-isif, postel} representing the address "postel@usc-isif"), or it falls off the end. When a pattern matches, the rule is reapplied until it fails.

The configuration file also supports the editing of addresses into different formats. For example, an address of the form:

    ucsfcgl!tef

might be mapped into:

    tef@ucsfcgl.UUCP

to conform to the domain syntax. Translations can also be done in the other direction.

3.7.5. Option setting

There are several options that can be set from the configuration file. These include the pathnames of various support files, timeouts, default modes, etc.

4. COMPARISON WITH OTHER MAILERS

4.1. Delivermail

Sendmail is an outgrowth of *delivermail*. The primary differences are:

(1) Configuration information is not compiled in. This change simplifies many of the problems of moving to other machines. It also allows easy debugging of new mailers.

(2) Address parsing is more flexible. For example, *delivermail* only supported one gateway to any network, whereas *sendmail* can be sensitive to host names and reroute to different gateways.

(3) Forwarding and :include: features eliminate the requirement that the system alias file be writable by any user (or that an update program be written, or that the system administration make all changes).

(4) *Sendmail* supports message batching across networks when a message is being sent to multiple recipients.

(5) A mail queue is provided in *sendmail.* Mail that cannot be delivered immediately but can potentially be delivered later is stored in this queue for a later retry. The queue also provides a buffer against system crashes; after the message has been collected it may be reliably redelivered even if the system crashes during the initial delivery.

(6) *Sendmail* uses the networking support provided by 4.2BSD to provide a direct interface networks such as the ARPANET and/or Ethernet using SMTP (the Simple Mail Transfer Protocol) over a TCP/IP connection.

4.2. MMDF

MMDF [Crocker79] spans a wider problem set than *sendmail.* For example, the domain of MMDF includes a "phone network" mailer, whereas *sendmail* calls on preexisting mailers in most cases.

MMDF and *sendmail* both support aliasing, customized mailers, message batching, automatic forwarding to gateways, queueing, and retransmission. MMDF supports two-stage timeout, which *sendmail* does not support.

The configuration for MMDF is compiled into the code[4].

Since MMDF does not consider backwards compatibility as a design goal, the address parsing is simpler but much less flexible.

It is somewhat harder to integrate a new channel[5] into MMDF. In particular, MMDF must know the location and format of host tables for all channels, and the channel must speak a special protocol. This allows MMDF to do additional verification (such as verifying host names) at submission time.

MMDF strictly separates the submission and delivery phases. Although *sendmail* has the concept of each of these stages, they are integrated into one program, whereas in MMDF they are split into two programs.

4.3. Message Processing Module

The Message Processing Module (MPM) discussed by Postel [Postel79b] matches *sendmail* closely in terms of its basic architecture. However, like MMDF, the MPM includes the network interface software as part of its domain.

MPM also postulates a duplex channel to the receiver, as does MMDF, thus allowing simpler handling of errors by the mailer than is possible in *sendmail.* When a message queued by *sendmail* is sent, any errors must be returned to the sender by the mailer itself. Both MPM and MMDF mailers can return an immediate error response, and a single error processor can create an appropriate response.

MPM prefers passing the message as a structured object, with type-length-value

[4]Dynamic configuration tables are currently being considered for MMDF; allowing the installer to select either compiled or dynamic tables.

[5]The MMDF equivalent of a *sendmail* "mailer."

tuples[6]. Such a convention requires a much higher degree of cooperation between mailers than is required by *sendmail.* MPM also assumes a universally agreed upon internet name space (with each address in the form of a net-host-user tuple), which *sendmail* does not.

5. EVALUATIONS AND FUTURE PLANS

Sendmail is designed to work in a nonhomogeneous environment. Every attempt is made to avoid imposing unnecessary constraints on the underlying mailers. This goal has driven much of the design. One of the major problems has been the lack of a uniform address space, as postulated in [Postel79a] and [Postel79b].

A nonuniform address space implies that a path will be specified in all addresses, either explicitly (as part of the address) or implicitly (as with implied forwarding to gateways). This restriction has the unpleasant effect of making replying to messages exceedingly difficult, since there is no one "address" for any person, but only a way to get there from wherever you are.

Interfacing to mail programs that were not initially intended to be applied in an internet environment has been amazingly successful, and has reduced the job to a manageable task.

Sendmail has knowledge of a few difficult environments built in. It generates ARPANET FTP/SMTP compatible error messages (prepended with three-digit numbers [Neigus73, Postel74, Postel82]) as necessary, optionally generates UNIX-style "From" lines on the front of messages for some mailers, and knows how to parse the same lines on input. Also, error handling has an option customized for BerkNet.

The decision to avoid doing any type of delivery where possible (even, or perhaps especially, local delivery) has turned out to be a good idea. Even with local delivery, there are issues of the location of the mailbox, the format of the mailbox, the locking protocol used, etc., that are best decided by other programs. One surprisingly major annoyance in many internet mailers is that the location and format of local mail is built in. The feeling seems to be that local mail is so common that it should be efficient. This feeling is not born out by our experience; on the contrary, the location and format of mailboxes seems to vary widely from system to system.

The ability to automatically generate a response to incoming mail (by forwarding mail to a program) seems useful ("I am on vacation until late August....") but can create problems such as forwarding loops (two people on vacation whose programs send notes back and forth, for instance) if these programs are not well written. A program could be written to do standard tasks correctly, but this would solve the general case.

It might be desirable to implement some form of load limiting. I am unaware of any mail system that addresses this problem, nor am I aware of any reasonable solution at this time.

The configuration file is currently practically inscrutable; considerable convenience could be realized with a higher-level format.

It seems clear that common protocols will be changing soon to accommodate changing requirements and environments. These changes will include modifications to the message header (e.g., [NBS80]) or to the body of the message itself (such as for multimedia messages [Postel80]). Experience indicates that these changes should be relatively trivial to integrate into the existing system.

In tightly coupled environments, it would be nice to have a name server such as Grapvine [Birrell82] integrated into the mail system. This would allow a site such as "Berkeley" to appear as a single host, rather than as a collection of hosts, and would allow people to move transparently among machines without having to change their addresses. Such a

[6]This is similar to the NBS standard.

facility would require an automatically updated database and some method of resolving conflicts. Ideally this would be effective even without all hosts being under a single management. However, it is not clear whether this feature should be integrated into the aliasing facility or should be considered a "value added" feature outside *sendmail* itself.

As a more interesting case, the CSNET name server [Solomon81] provides an facility that goes beyond a single tightly-coupled environment. Such a facility would normally exist outside of *sendmail* however.

ACKNOWLEDGEMENTS

Thanks are due to Kurt Shoens for his continual cheerful assistance and good advice, Bill Joy for pointing me in the correct direction (over and over), and Mark Horton for more advice, prodding, and many of the good ideas. Kurt and Eric Schmidt are to be credited for using *delivermail* as a server for their programs (*Mail* and BerkNet respectively) before any sane person should have, and making the necessary modifications promptly and happily. Eric gave me considerable advice about the perils of network software which saved me an unknown amount of work and grief. Mark did the original implementation of the DBM version of aliasing, installed the VFORK code, wrote the current version of *rmail*, and was the person who really convinced me to put the work into *delivermail* to turn it into *sendmail*. Kurt deserves accolades for using *sendmail* when I was myself afraid to take the risk; how a person can continue to be so enthusiastic in the face of so much bitter reality is beyond me.

Kurt, Mark, Kirk McKusick, Marvin Solomon, and many others have reviewed this paper, giving considerable useful advice.

Special thanks are reserved for Mike Stonebraker at Berkeley and Bob Epstein at Britton-Lee, who both knowingly allowed me to put so much work into this project when there were so many other things I really should have been working on.

REFERENCES

[Birrell82]	Birrell, A. D., Levin, R., Needham, R. M., and Schroeder, M. D., "Grapevine: An Exercise in Distributed Computing." In *Comm. A.C.M. 25,* 4, April 82.
[Borden79]	Borden, S., Gaines, R. S., and Shapiro, N. Z., *The MH Message Handling System: Users' Manual.* R-2367-PAF. Rand Corporation. October 1979.
[Crocker77a]	Crocker, D. H., Vittal, J. J., Pogran, K. T., and Henderson, D. A. Jr., *Standard for the Format of ARPA Network Text Messages.* RFC 733, NIC 41952. In [Feinler78]. November 1977.
[Crocker77b]	Crocker, D. H., *Framework and Functions of the MS Personal Message System.* R-2134-ARPA, Rand Corporation, Santa Monica, California. 1977.
[Crocker79]	Crocker, D. H., Szurkowski, E. S., and Farber, D. J., *An Internetwork Memo Distribution Facility — MMDF.* 6th Data Communication Symposium, Asilomar. November 1979.
[Crocker82]	Crocker, D. H., *Standard for the Format of Arpa Internet Text Messages.* RFC 822. Network Information Center, SRI International, Menlo Park, California. August 1982.
[Metcalfe76]	Metcalfe, R., and Boggs, D., "Ethernet: Distributed Packet Switching for Local Computer Networks", *Communications of the ACM 19,* 7. July 1976.
[Feinler78]	Feinler, E., and Postel, J. (eds.), *ARPANET Protocol Handbook.* NIC 7104, Network Information Center, SRI International, Menlo Park, California. 1978.
[NBS80]	National Bureau of Standards, *Specification of a Draft Message Format Standard.* Report No. ICST/CBOS 80-2. October 1980.
[Neigus73]	Neigus, N., *File Transfer Protocol for the ARPA Network.* RFC 542, NIC 17759. In [Feinler78]. August, 1973.
[Nowitz78a]	Nowitz, D. A., and Lesk, M. E., *A Dial-Up Network of UNIX Systems.* Bell Laboratories. In UNIX Programmer's Manual, Seventh Edition, Volume 2. August, 1978.
[Nowitz78b]	Nowitz, D. A., *Uucp Implementation Description.* Bell Laboratories. In UNIX Programmer's Manual, Seventh Edition, Volume 2. October, 1978.
[Postel74]	Postel, J., and Neigus, N., Revised FTP Reply Codes. RFC 640, NIC 30843. In [Feinler78]. June, 1974.
[Postel77]	Postel, J., *Mail Protocol.* NIC 29588. In [Feinler78]. November 1977.
[Postel79a]	Postel, J., *Internet Message Protocol.* RFC 753, IEN 85. Network Information Center, SRI International, Menlo Park, California. March 1979.
[Postel79b]	Postel, J. B., *An Internetwork Message Structure.* In *Proceedings of the Sixth Data Communications Symposium,* IEEE. New York. November 1979.
[Postel80]	Postel, J. B., *A Structured Format for Transmission of Multi-Media Documents.* RFC 767. Network Information Center, SRI International, Menlo Park, California. August 1980.

[Postel82]	Postel, J. B., *Simple Mail Transfer Protocol.* RFC821 (obsoleting RFC788). Network Information Center, SRI International, Menlo Park, California. August 1982.
[Schmidt79]	Schmidt, E., *An Introduction to the Berkeley Network.* University of California, Berkeley California. 1979.
[Shoens79]	Shoens, K., *Mail Reference Manual.* University of California, Berkeley. In UNIX Programmer's Manual, Seventh Edition, Volume 2C. December 1979.
[Sluizer81]	Sluizer, S., and Postel, J. B., *Mail Transfer Protocol.* RFC 780. Network Information Center, SRI International, Menlo Park, California. May 1981.
[Solomon81]	Solomon, M., Landweber, L., and Neuhengen, D., "The Design of the CSNET Name Server." CS-DN-2, University of Wisconsin, Madison. November 1981.
[Su82]	Su, Zaw-Sing, and Postel, Jon, *The Domain Naming Convention for Internet User Applications.* RFC819. Network Information Center, SRI International, Menlo Park, California. August 1982.
[UNIX83]	*The UNIX Programmer's Manual, Seventh Edition,* Virtual VAX-11 Version, Volume 1. Bell Laboratories, modified by the University of California, Berkeley, California. March, 1983.

SENDMAIL

INSTALLATION AND OPERATION GUIDE

Eric Allman
Britton-Lee, Inc.

Version 4.2

TABLE OF CONTENTS

SENDMAIL

INSTALLATION AND OPERATION GUIDE

Eric Allman
Britton-Lee, Inc.

Version 4.2

Sendmail implements a general purpose internetwork mail routing facility under the UNIX* operating system. It is not tied to any one transport protocol — its function may be likened to a crossbar switch, relaying messages from one domain into another. In the process, it can do a limited amount of message header editing to put the message into a format that is appropriate for the receiving domain. All of this is done under the control of a configuration file.

Due to the requirements of flexibility for *sendmail*, the configuration file can seem somewhat unapproachable. However, there are only a few basic configurations for most sites, for which standard configuration files have been supplied. Most other configurations can be built by adjusting an existing configuration files incrementally.

Although *sendmail* is intended to run without the need for monitoring, it has a number of features that may be used to monitor or adjust the operation under unusual circumstances. These features are described.

Section one describes how to do a basic *sendmail* installation. Section two explains the day-to-day information you should know to maintain your mail system. If you have a relatively normal site, these two sections should contain sufficient information for you to install *sendmail* and keep it happy. Section three describes some parameters that may be safely tweaked. Section four has information regarding the command line arguments. Section five contains the nitty-gritty information about the configuration file. This section is for masochists and people who must write their own configuration file. The appendixes give a brief but detailed explanation of a number of features not described in the rest of the paper.

The references in this paper are actually found in the companion paper *Sendmail — An Internetwork Mail Router*. This other paper should be read before this manual to gain a basic understanding of how the pieces fit together.

1. BASIC INSTALLATION

There are two basic steps to installing sendmail. The hard part is to build the configuration table. This is a file that sendmail reads when it starts up that describes the mailers it knows about, how to parse addresses, how to rewrite the message header, and the settings of various options. Although the configuration table is quite complex, a configuration can usually be built by adjusting an existing off-the-shelf configuration. The second part is actually doing the installation, i.e., creating the necessary files, etc.

The remainder of this section will describe the installation of sendmail assuming you can use one of the existing configurations and that the standard installation parameters are acceptable. All pathnames and examples are given from the root of the *sendmail* subtree.

*UNIX is a trademark of Bell Laboratories.

1.1. Off-The-Shelf Configurations

The configuration files are all in the subdirectory *cf* of the sendmail directory. The ones used at Berkeley are in *m4*(1) format; files with names ending ".m4" are *m4* include files, while files with names ending ".mc" are the master files. Files with names ending ".cf" are the *m4* processed versions of the corresponding ".mc" file.

Two off the shelf configuration files are supplied to handle the basic cases: *cf/arpaproto.cf* for Arpanet (TCP) sites and *cf/uucpproto.cf* for UUCP sites. These are *not* in *m4* format. The file you need should be copied to a file with the same name as your system, e.g.,

```
cp uucpproto.cf ucsfcgl.cf
```

This file is now ready for installation as */usr/lib/sendmail.cf.*

1.2. Installation Using the Makefile

A makefile exists in the root of the *sendmail* directory that will do all of these steps for a 4.2BSD system. It may have to be slightly tailored for use on other systems.

Before using this makefile, you should already have created your configuration file and left it in the file "cf/*system*.cf" where *system* is the name of your system (i.e., what is returned by *hostname*(1)). If you do not have *hostname* you can use the declaration "HOST=*system*" on the *make*(1) command line. You should also examine the file *md/config.m4* and change the *m4* macros there to reflect any libraries and compilation flags you may need.

The basic installation procedure is to type:

```
make
make install
```

in the root directory of the *sendmail* distribution. This will make all binaries and install them in the standard places. The second *make* command must be executed as the superuser (root).

1.3. Installation by Hand

Along with building a configuration file, you will have to install the *sendmail* startup into your UNIX system. If you are doing this installation in conjunction with a regular Berkeley UNIX install, these steps will already be complete. Many of these steps will have to be executed as the superuser (root).

1.3.1. lib/libsys.a

The library in lib/libsys.a contains some routines that should in some sense be part of the system library. These are the system logging routines and the new directory access routines (if required). If you are not running the new 4.2BSD directory code and do not have the compatibility routines installed in your system library, you should execute the commands:

```
cd lib
make ndir
```

This will compile and install the 4.2 compatibility routines in the library. You should then type:

```
cd lib     # if required
make
```

This will recompile and fill the library.

1.3.2. /usr/lib/sendmail

The binary for sendmail is located in /usr/lib. There is a version available in the source directory that is probably inadequate for your system. You should plan on recompiling and installing the entire system:

```
cd src
rm −f *.o
make
cp sendmail /usr/lib
```

1.3.3. /usr/lib/sendmail.cf

The configuration file that you created earlier should be installed in /usr/lib/sendmail.cf:

cp cf/*system*.cf /usr/lib/sendmail.cf

1.3.4. /usr/ucb/newaliases

If you are running delivermail, it is critical that the *newaliases* command be replaced. This can just be a link to *sendmail*:

```
rm −f /usr/ucb/newaliases
ln /usr/lib/sendmail /usr/ucb/newaliases
```

1.3.5. /usr/lib/sendmail.cf

The configuration file must be installed in /usr/lib. This is described above.

1.3.6. /usr/spool/mqueue

The directory */usr/spool/mqueue* should be created to hold the mail queue. This directory should be mode 777 unless *sendmail* is run setuid, when *mqueue* should be owned by the sendmail owner and mode 755.

1.3.7. /usr/lib/aliases*

The system aliases are held in three files. The file "/usr/lib/aliases" is the master copy. A sample is given in "lib/aliases" which includes some aliases which *must* be defined:

```
cp lib/aliases /usr/lib/aliases
```

You should extend this file with any aliases that are apropos to your system.

Normally *sendmail* looks at a version of these files maintained by the *dbm*(3) routines. These are stored in "/usr/lib/aliases.dir" and "/usr/lib/aliases.pag." These can initially be created as empty files, but they will have to be initialized promptly. These should be mode 666 if you are running a reasonably relaxed system:

```
cp /dev/null /usr/lib/aliases.dir
cp /dev/null /usr/lib/aliases.pag
chmod 666 /usr/lib/aliases.*
newaliases
```

1.3.8. /usr/lib/sendmail.fc

If you intend to install the frozen version of the configuration file (for quick startup) you should create the file /usr/lib/sendmail.fc and initialize it. This step may be safely skipped.

```
cp /dev/null /usr/lib/sendmail.fc
/usr/lib/sendmail −bz
```

1.3.9. /etc/rc

It will be necessary to start up the sendmail daemon when your system reboots. This daemon performs two functions: it listens on the SMTP socket for connections (to receive mail from a remote system) and it processes the queue periodically to insure that mail gets delivered when hosts come up.

Add the following lines to "/etc/rc" (or "/etc/rc.local" as appropriate) in the area where it is starting up the daemons:

```
if [ −f /usr/lib/sendmail ]; then
        (cd /usr/spool/mqueue; rm −f [lnx]f*)
        /usr/lib/sendmail −bd −q30m &
        echo −n ' sendmail' >/dev/console
fi
```

The "cd" and "rm" commands insure that all lock files have been removed; extraneous lock files may be left around if the system goes down in the middle of processing a message. The line that actually invokes *sendmail* has two flags: "−bd" causes it to listen on the SMTP port, and "−q30m" causes it to run the queue every half hour.

If you are not running a version of UNIX that supports Berkeley TCP/IP, do not include the **−bd** flag.

1.3.10. /usr/lib/sendmail.hf

This is the help file used by the SMTP HELP command. It should be copied from "lib/sendmail.hf":

```
cp lib/sendmail.hf /usr/lib
```

1.3.11. /usr/lib/sendmail.st

If you wish to collect statistics about your mail traffic, you should create the file "/usr/lib/sendmail.st":

```
cp /dev/null /usr/lib/sendmail.st
chmod 666 /usr/lib/sendmail.st
```

This file does not grow. It is printed with the program "aux/mailstats."

1.3.12. /etc/syslog

You may want to run the *syslog* program (to collect log information about sendmail). This program normally resides in */etc/syslog*, with support files */etc/syslog.conf* and */etc/syslog.pid*. The program is located in the *aux* subdirectory of the *sendmail* distribution. The file */etc/syslog.conf* describes the file(s) that sendmail will log in. For a complete description of syslog, see the manual page for *syslog*(8) (located in *sendmail/doc* on the distribution).

1.3.13. /usr/ucb/newaliases

If *sendmail* is invoked as "newaliases," it will simulate the **−bi** flag (i.e., will rebuild the alias database; see below). This should be a link to /usr/lib/sendmail.

1.3.14. /usr/ucb/mailq

If *sendmail* is invoked as "mailq," it will simulate the **−bp** flag (i.e., *sendmail* will print the contents of the mail queue; see below). This should be a link to /usr/lib/sendmail.

2. NORMAL OPERATIONS

2.1. Quick Configuration Startup

A fast version of the configuration file may be set up by using the **−bz** flag:

```
/usr/lib/sendmail −bz
```

This creates the file */usr/lib/sendmail.fc* ("frozen configuration"). This file is an image of *sendmail's* data space after reading in the configuration file. If this file exists, it is used instead of */usr/lib/sendmail.cf sendmail.fc* must be rebuilt manually every time *sendmail.cf* is changed.

The frozen configuration file will be ignored if a −C flag is specified or if sendmail detects that it is out of date. However, the heuristics are not strong so this should not be trusted.

2.2. The System Log

The system log is supported by the *syslog*(8) program.

2.2.1. Format

Each line in the system log consists of a timestamp, the name of the machine that generated it (for logging from several machines over the ethernet), the word "sendmail:", and a message.

2.2.2. Levels

If you have *syslog*(8) or an equivalent installed, you will be able to do logging. There is a large amount of information that can be logged. The log is arranged as a succession of levels. At the lowest level only extremely strange situations are logged. At the highest level, even the most mundane and uninteresting events are recorded for posterity. As a convention, log levels under ten are considered "useful;" log levels above ten are usually for debugging purposes.

A complete description of the log levels is given in section 4.3.

2.3. The Mail Queue

The mail queue should be processed transparently. However, you may find that manual intervention is sometimes necessary. For example, if a major host is down for a period of time the queue may become clogged. Although sendmail ought to recover gracefully when the host comes up, you may find performance unacceptably bad in the meantime.

2.3.1. Printing the queue

The contents of the queue can be printed using the *mailq* command (or by specifying the **−bp** flag to sendmail):

```
mailq
```

This will produce a listing of the queue id's, the size of the message, the date the message entered the queue, and the sender and recipients.

2.3.2. Format of queue files

All queue files have the form *x*f*AA99999* where *AA99999* is the *id* for this file and the *x* is a type. The types are:

d The data file. The message body (excluding the header) is kept in this file.

l The lock file. If this file exists, the job is currently being processed, and a queue run will not process the file. For that reason, an extraneous lf file can cause a job to apparently disappear (it will not even time out!).

n This file is created when an id is being created. It is a separate file to insure that no mail can ever be destroyed due to a race condition. It should exist for no more than a few milliseconds at any given time.

q The queue control file. This file contains the information necessary to process the job.

t A temporary file. These are an image of the **qf** file when it is being rebuilt. It should be renamed to a **qf** file very quickly.

x A transcript file, existing during the life of a session showing everything that happens during that session.

The **qf** file is structured as a series of lines each beginning with a code letter. The lines are as follows:

D The name of the data file. There may only be one of these lines.

H A header definition. There may be any number of these lines. The order is important: they represent the order in the final message. These use the same syntax as header definitions in the configuration file.

R A recipient address. This will normally be completely aliased, but is actually realiased when the job is processed. There will be one line for each recipient.

S The sender address. There may only be one of these lines.

T The job creation time. This is used to compute when to time out the job.

P The current message priority. This is used to order the queue. Higher numbers mean lower priorities. The priority increases as the message sits in the queue. The initial priority depends on the message class and the size of the message.

M A message. This line is printed by the *mailq* command, and is generally used to store status information. It can contain any text.

As an example, the following is a queue file sent to "mckusick@calder" and "wnj":

```
DdfA13557
Seric
T404261372
P132
Rmckusick@calder
Rwnj
H?D?date: 23-Oct-82 15:49:32-PDT (Sat)
H?F?from: eric (Eric Allman)
H?x?full-name: Eric Allman
Hsubject: this is an example message
Hmessage-id: <8209232249.13557@UCBARPA.BERKELEY.ARPA>
Hreceived: by UCBARPA.BERKELEY.ARPA (3.227 [10/22/82])
        id A13557; 23-Oct-82 15:49:32-PDT (Sat)
Hphone: (415) 548-3211
HTo: mckusick@calder, wnj
```

This shows the name of the data file, the person who sent the message, the submission time (in seconds since January 1, 1970), the message priority, the message class, the recipients, and the headers for the message.

2.3.3. Forcing the queue

Sendmail should run the queue automatically at intervals. The algorithm is to read and sort the queue, and then to attempt to process all jobs in order. When it attempts to run the job, *sendmail* first checks to see if the job is locked. If so, it ignores the job.

There is no attempt to insure that only one queue processor exists at any time, since there is no guarantee that a job cannot take forever to process. Due to the locking algorithm, it is impossible for one job to freeze the queue. However, an uncooperative recipient host or a program recipient that never returns can accumulate many processes in your system. Unfortunately, there is no way to resolve this without violating the protocol.

In some cases, you may find that a major host going down for a couple of days may create a prohibitively large queue. This will result in *sendmail* spending an inordinate amount of time sorting the queue. This situation can be fixed by moving the queue to a temporary place and creating a new queue. The old queue can be run later when the offending host returns to service.

To do this, it is acceptable to move the entire queue directory:

```
cd /usr/spool
mv mqueue omqueue; mkdir mqueue; chmod 777 mqueue
```

You should then kill the existing daemon (since it will still be processing in the old queue directory) and create a new daemon.

To run the old mail queue, run the following command:

```
/usr/lib/sendmail −oQ/usr/spool/omqueue −q
```

The **−oQ** flag specifies an alternate queue directory and the **−q** flag says to just run every job in the queue. If you have a tendency toward voyeurism, you can use the **−v** flag to watch what is going on.

When the queue is finally emptied, you can remove the directory:

```
rmdir /usr/spool/omqueue
```

2.4. The Alias Database

The alias database exists in two forms. One is a text form, maintained in the file */usr/lib/aliases.* The aliases are of the form

```
name: name1, name2, ...
```

Only local names may be aliased; e.g.,

```
eric@mit-xx: eric@berkeley
```

will not have the desired effect. Aliases may be continued by starting any continuation lines with a space or a tab. Blank lines and lines beginning with a sharp sign ("#") are comments.

The second form is processed by the *dbm*(3) library. This form is in the files */usr/lib/aliases.dir* and */usr/lib/aliases.pag.* This is the form that *sendmail* actually uses to resolve aliases. This technique is used to improve performance.

2.4.1. Rebuilding the alias database

The DBM version of the database may be rebuilt explicitly by executing the command

```
newaliases
```

This is equivalent to giving *sendmail* the **−bi** flag:

```
/usr/lib/sendmail −bi
```

If the "D" option is specified in the configuration, *sendmail* will rebuild the alias database automatically if possible when it is out of date. The conditions under which it will do this are:

(1) The DBM version of the database is mode 666. -or-

(2) *Sendmail* is running setuid to root.

Auto-rebuild can be dangerous on heavily loaded machines with large alias files; if it might take more than five minutes to rebuild the database, there is a chance that several processes will start the rebuild process simultaneously.

2.4.2. Potential problems

There are a number of problems that can occur with the alias database. They all result from a *sendmail* process accessing the DBM version while it is only partially built. This can happen under two circumstances: One process accesses the database while another process is rebuilding it, or the process rebuilding the database dies (due to being killed or a system crash) before completing the rebuild.

Sendmail has two techniques to try to relieve these problems. First, it ignores interrupts while rebuilding the database; this avoids the problem of someone aborting the process leaving a partially rebuilt database. Second, at the end of the rebuild it adds an alias of the form

```
@: @
```

(which is not normally legal). Before sendmail will access the database, it checks to insure that this entry exists[1]. It will wait up to five minutes for this entry to appear, at which point it will force a rebuild itself[2].

2.4.3. List owners

If an error occurs on sending to a certain address, say "*x*", *sendmail* will look for an alias of the form "owner-*x*" to receive the errors. This is typically useful for a mailing list where the submitter of the list has no control over the maintanence of the list itself; in this case the list maintainer would be the owner of the list. For example:

```
unix-wizards: eric@ucbarpa, wnj@monet, nosuchuser,
        sam@matisse
owner-unix-wizards: eric@ucbarpa
```

would cause "eric@ucbarpa" to get the error that will occur when someone sends to unix-wizards due to the inclusion of "nosuchuser" on the list.

2.5. Per-User Forwarding (.forward Files)

As an alternative to the alias database, any user may put a file with the name ".forward" in his or her home directory. If this file exists, *sendmail* redirects mail for that user to the list of addresses listed in the .forward file. For example, if the home directory for user "mckusick" has a .forward file with contents:

```
mckusick@ernie
kirk@calder
```

then any mail arriving for "mckusick" will be redirected to the specified accounts.

2.6. Special Header Lines

Several header lines have special interpretations defined by the configuration file. Others have interpretations built into *sendmail* that cannot be changed without changing the code. These builtins are described here.

[1]The "a" option is required in the configuration for this action to occur. This should normally be specified unless you are running *delivermail* in parallel with *sendmail*.

[2]Note: the "D" option must be specified in the configuration file for this operation to occur.

2.6.1. Return-Receipt-To:

If this header is sent, a message will be sent to any specified addresses when the final delivery is complete. if the mailer has the **l** flag (local delivery) set in the mailer descriptor.

2.6.2. Errors-To:

If errors occur anywhere during processing, this header will cause error messages to go to the listed addresses rather than to the sender. This is intended for mailing lists.

2.6.3. Apparently-To:

If a message comes in with no recipients listed in the message (in a To:, Cc:, or Bcc: line) then *sendmail* will add an "Apparently-To:" header line for any recipients it is aware of. This is not put in as a standard recipient line to warn any recipients that the list is not complete.

At least one recipient line is required under RFC 822.

3. ARGUMENTS

The complete list of arguments to *sendmail* is described in detail in Appendix A. Some important arguments are described here.

3.1. Queue Interval

The amount of time between forking a process to run through the queue is defined by the **−q** flag. If you run in mode **f** or **a** this can be relatively large, since it will only be relevant when a host that was down comes back up. If you run in **q** mode it should be relatively short, since it defines the maximum amount of time that a message may sit in the queue.

3.2. Daemon Mode

If you allow incoming mail over an IPC connection, you should have a daemon running. This should be set by your */etc/rc* file using the **−bd** flag. The **−bd** flag and the **−q** flag may be combined in one call:

```
/usr/lib/sendmail −bd −q30m
```

3.3. Forcing the Queue

In some cases you may find that the queue has gotten clogged for some reason. You can force a queue run using the **−q** flag (with no value). It is entertaining to use the **−v** flag (verbose) when this is done to watch what happens:

```
/usr/lib/sendmail −q −v
```

3.4. Debugging

There are a fairly large number of debug flags built into *sendmail*. Each debug flag has a number and a level, where higher levels means to print out more information. The convention is that levels greater than nine are "absurd," i.e., they print out so much information that you wouldn't normally want to see them except for debugging that particular piece of code. Debug flags are set using the **−d** option; the syntax is:

debug-flag:	**−d** debug-list
debug-list:	debug-option [, debug-option]
debug-option:	debug-range [. debug-level]
debug-range:	integer | integer − integer
debug-level:	integer

where spaces are for reading ease only. For example,

−d12	Set flag 12 to level 1
−d12.3	Set flag 12 to level 3
−d3-17	Set flags 3 through 17 to level 1
−d3-17.4	Set flags 3 through 17 to level 4

For a complete list of the available debug flags you will have to look at the code (they are too dynamic to keep this documentation up to date).

3.5. Trying a Different Configuration File

An alternative configuration file can be specified using the **−C** flag; for example,

```
/usr/lib/sendmail −Ctest.cf
```

uses the configuration file *test.cf* instead of the default */usr/lib/sendmail.cf.* If the **−C** flag has no value it defaults to *sendmail.cf* in the current directory.

3.6. Changing the Values of Options

Options can be overridden using the **−o** flag. For example,

```
/usr/lib/sendmail −oT2m
```

sets the T (timeout) option to two minutes for this run only.

4. TUNING

There are a number of configuration parameters you may want to change, depending on the requirements of your site. Most of these are set using an option in the configuration file. For example, the line "OT3d" sets option "T" to the value "3d" (three days).

4.1. Timeouts

All time intervals are set using a scaled syntax. For example, "10m" represents ten minutes, whereas "2h30m" represents two and a half hours. The full set of scales is:

s	seconds
m	minutes
h	hours
d	days
w	weeks

4.1.1. Queue interval

The argument to the **−q** flag specifies how often a subdaemon will run the queue. This is typically set to between five minutes and one half hour.

4.1.2. Read timeouts

It is possible to time out when reading the standard input or when reading from a remote SMTP server. Technically, this is not acceptable within the published protocols. However, it might be appropriate to set it to something large in certain environments (such as an hour). This will reduce the chance of large numbers of idle daemons piling up on your system. This timeout is set using the **r** option in the configuration file.

4.1.3. Message timeouts

After sitting in the queue for a few days, a message will time out. This is to insure that at least the sender is aware of the inability to send a message. The timeout is typically set to three days. This timeout is set using the **T** option in the configuration file.

The time of submission is set in the queue, rather than the amount of time left until timeout. As a result, you can flush messages that have been hanging for a short period by running the queue with a short message timeout. For example,

```
/usr/lib/sendmail −oT1d −q
```

will run the queue and flush anything that is one day old.

4.2. Delivery Mode

There are a number of delivery modes that *sendmail* can operate in, set by the "d" configuration option. These modes specify how quickly mail will be delivered. Legal modes are:

i	deliver interactively (synchronously)
b	deliver in background (asynchronously)
q	queue only (don't deliver)

There are tradeoffs. Mode "i" passes the maximum amount of information to the sender, but is hardly ever necessary. Mode "q" puts the minimum load on your machine, but means that delivery may be delayed for up to the queue interval. Mode "b" is probably a good compromise. However, this mode can cause large numbers of processes if you have a mailer that takes a long time to deliver a message.

4.3. Log Level

The level of logging can be set for sendmail. The default using a standard configuration table is level 9. The levels are as follows:

0	No logging.
1	Major problems only.
2	Message collections and failed deliveries.
3	Successful deliveries.
4	Messages being defered (due to a host being down, etc.).
5	Normal message queueups.
6	Unusual but benign incidents, e.g., trying to process a locked queue file.
9	Log internal queue id to external message id mappings. This can be useful for tracing a message as it travels between several hosts.
12	Several messages that are basically only of interest when debugging.
16	Verbose information regarding the queue.

4.4. File Modes

There are a number of files that may have a number of modes. The modes depend on what functionality you want and the level of security you require.

4.4.1. To suid or not to suid?

Sendmail can safely be made setuid to root. At the point where it is about to *exec*(2) a mailer, it checks to see if the userid is zero; if so, it resets the userid and groupid to a default (set by the **u** and **g** options). (This can be overridden by setting the S flag to the mailer for mailers that are trusted and must be called as root.)

However, this will cause mail processing to be accounted (using *sa*(8)) to root rather than to the user sending the mail.

4.4.2. Temporary file modes

The mode of all temporary files that *sendmail* creates is determined by the "F" option. Reasonable values for this option are 0600 and 0644. If the more permissive mode is selected, it will not be necessary to run *sendmail* as root at all (even when running the queue).

4.4.3. Should my alias database be writable?

At Berkeley we have the alias database (/usr/lib/aliases*) mode 666. There are some dangers inherent in this approach: any user can add him-/her-self to any list, or can "steal" any other user's mail. However, we have found users to be basically trustworthy, and the cost of having a read-only database greater than the expense of finding and eradicating the rare nasty person.

The database that *sendmail* actually used is represented by the two files *aliases.dir* and *aliases.pag* (both in /usr/lib). The mode on these files should match the mode on /usr/lib/aliases. If *aliases* is writable and the DBM files (*aliases.dir* and *aliases.pag*) are not, users will be unable to reflect their desired changes through to the actual database. However, if *aliases* is read-only and the DBM files are writable, a slightly sophisticated user can arrange to steal mail anyway.

If your DBM files are not writable by the world or you do not have auto-rebuild enabled (with the "D" option), then you must be careful to reconstruct the alias database each time you change the text version:

```
newaliases
```

If this step is ignored or forgotten any intended changes will also be ignored or forgotten.

5. THE WHOLE SCOOP ON THE CONFIGURATION FILE

This section describes the configuration file in detail, including hints on how to write one of your own if you have to.

There is one point that should be made clear immediately: the syntax of the configuration file is designed to be reasonably easy to parse, since this is done every time *sendmail* starts up, rather than easy for a human to read or write. On the "future project" list is a configuration-file compiler.

An overview of the configuration file is given first, followed by details of the semantics.

5.1. The Syntax

The configuration file is organized as a series of lines, each of which begins with a single character defining the semantics for the rest of the line. Lines beginning with a space or a tab are continuation lines (although the semantics are not well defined in many places). Blank lines and lines beginning with a sharp symbol ('#') are comments.

5.1.1. R and S — rewriting rules

The core of address parsing are the rewriting rules. These are an ordered production system. *Sendmail* scans through the set of rewriting rules looking for a match on the left hand side (LHS) of the rule. When a rule matches, the address is replaced by the right hand side (RHS) of the rule.

There are several sets of rewriting rules. Some of the rewriting sets are used internally and must have specific semantics. Other rewriting sets do not have specifically assigned semantics, and may be referenced by the mailer definitions or by other rewriting sets.

The syntax of these two commands are:

S*n*

Sets the current ruleset being collected to *n*. If you begin a ruleset more than once it deletes the old definition.

R*lhs rhs comments*

The fields must be separated by at least one tab character; there may be embedded spaces in the fields. The *lhs* is a pattern that is applied to the input. If it matches, the input is rewritten to the *rhs*. The *comments* are ignored.

5.1.2. D — define macro

Macros are named with a single character. These may be selected from the entire ASCII set, but user-defined macros should be selected from the set of upper case letters only. Lower case letters and special symbols are used internally.

The syntax for macro definitions is:

D*x val*

where *x* is the name of the macro and *val* is the value it should have. Macros can be interpolated in most places using the escape sequence **$***x*.

5.1.3. C and F — define classes

Classes of words may be defined to match on the left hand side of rewriting rules. For example a class of all local names for this site might be created so that attempts to send to oneself can be eliminated. These can either be defined directly in the configuration file or read in from another file. Classes may be given names from the set of upper case letters. Lower case letters and special characters are reserved for system use.

The syntax is:

C*c word1 word2...*
F*cfile* [*format*]

The first form defines the class *c* to match any of the named words. It is permissible to split them among multiple lines; for example, the two forms:

```
CHmonet ucbmonet
```

and

```
CHmonet
CHucbmonet
```

are equivalent. The second form reads the elements of the class *c* from the named *file*; the *format* is a *scanf*(3) pattern that should produce a single string.

5.1.4. M — define mailer

Programs and interfaces to mailers are defined in this line. The format is:

M*name*, {*field*=*value*}*

where *name* is the name of the mailer (used internally only) and the "field=name" pairs define attributes of the mailer. Fields are:

Path	The pathname of the mailer
Flags	Special flags for this mailer
Sender	A rewriting set for sender addresses
Recipient	A rewriting set for recipient addresses
Argv	An argument vector to pass to this mailer
Eol	The end-of-line string for this mailer
Maxsize	The maximum message length to this mailer

Only the first character of the field name is checked.

5.1.5. H — define header

The format of the header lines that sendmail inserts into the message are defined by the **H** line. The syntax of this line is:

H[**?***mflags***?**]*hname*: *htemplate*

Continuation lines in this spec are reflected directly into the outgoing message. The *htemplate* is macro expanded before insertion into the message. If the *mflags* (surrounded by question marks) are specified, at least one of the specified flags must be stated in the mailer definition for this header to be automatically output. If one of these headers is in the input it is reflected to the output regardless of these flags.

Some headers have special semantics that will be described below.

5.1.6. O — set option

There are a number of "random" options that can be set from a configuration file. Options are represented by single characters. The syntax of this line is:

O*o value*

This sets option *o* to be *value*. Depending on the option, *value* may be a string, an integer, a boolean (with legal values "t", "T", "f", or "F"; the default is TRUE), or a time interval.

5.1.7. T — define trusted users

Trusted users are those users who are permitted to override the sender address using the **−f** flag. These typically are "root," "uucp," and "network," but on some users it may be convenient to extend this list to include other users, perhaps to support a separate UUCP login for each host. The syntax of this line is:

T*user1 user2...*

There may be more than one of these lines.

5.1.8. P — precedence definitions

Values for the "Precedence:" field may be defined using the **P** control line. The syntax of this field is:

P*name***=***num*

When the *name* is found in a "Precedence:" field, the message class is set to *num*. Higher numbers mean higher precedence. Numbers less than zero have the special property that error messages will not be returned. The default precedence is zero. For example, our list of precedences is:

```
Pfirst-class=0
Pspecial-delivery=100
Pjunk=-100
```

5.2. The Semantics

This section describes the semantics of the configuration file.

5.2.1. Special macros, conditionals

Macros are interpolated using the construct **$***x*, where *x* is the name of the macro to be interpolated. In particular, lower case letters are reserved to have special semantics, used to pass information in or out of sendmail, and some special characters are reserved to provide conditionals, etc.

The following macros *must* be defined to transmit information into *sendmail:*

e	The SMTP entry message
j	The "official" domain name for this site
l	The format of the UNIX from line
n	The name of the daemon (for error messages)
o	The set of "operators" in addresses
q	default format of sender address

The **$e** macro is printed out when SMTP starts up. The first word must be the **$j** macro. The **$j** macro should be in RFC821 format. The **$l** and **$n** macros can be considered constants except under terribly unusual circumstances. The **$o** macro consists of a list of characters which will be considered tokens and which will separate tokens when doing parsing. For example, if "r" were in the **$o** macro, then the input "address" would be scanned as three tokens: "add," "r," and "ess." Finally, the **$q** macro specifies how an address should appear in a message when it is defaulted. For example, on our system these definitions are:

```
De$j Sendmail $v ready at $b
DnMAILER-DAEMON
DlFrom $g  $d
Do.:%@!^=/
Dq$g$?x ($x)$.
Dj$H.$D
```

An acceptable alternative for the **$q** macro is "`$?x$x $.<$g>`". These correspond to the following two formats:

```
eric@Berkeley (Eric Allman)
Eric Allman <eric@Berkeley>
```

Some macros are defined by *sendmail* for interpolation into argv's for mailers or for other contexts. These macros are:

a	The origination date in Arpanet format
b	The current date in Arpanet format
c	The hop count
d	The date in UNIX (ctime) format
f	The sender (from) address
g	The sender address relative to the recipient
h	The recipient host
i	The queue id
p	Sendmail's pid
r	Protocol used
s	Sender's host name
t	A numeric representation of the current time
u	The recipient user
v	The version number of sendmail
w	The hostname of this site
x	The full name of the sender
y	The id of the sender's tty
z	The home directory of the recipient

There are three types of dates that can be used. The **$a** and **$b** macros are in Arpanet format; **$a** is the time as extracted from the "Date:" line of the message (if there was one), and **$b** is the current date and time (used for postmarks). If no "Date:" line is found in the incoming message, **$a** is set to the current time also. The **$d** macro is equivalent to the **$a** macro in UNIX (ctime) format.

The **$f** macro is the id of the sender as originally determined; when mailing to a specific host the **$g** macro is set to the address of the sender *relative to the recipient.* For example, if I send to "bollard@matisse" from the machine "ucbarpa" the **$f** macro will be "eric" and the **$g** macro will be "eric@ucbarpa."

The **$x** macro is set to the full name of the sender. This can be determined in several ways. It can be passed as flag to *sendmail.* The second choice is the value of the "Full-name:" line in the header if it exists, and the third choice is the comment field of a "From:" line. If all of these fail, and if the message is being originated locally, the full name is looked up in the */etc/passwd* file.

When sending, the **$h**, **$u**, and **$z** macros get set to the host, user, and home directory (if local) of the recipient. The first two are set from the **$@** and **$:** part of the rewriting rules, respectively.

The **$p** and **$t** macros are used to create unique strings (e.g., for the "Message-Id:" field). The **$i** macro is set to the queue id on this host; if put into the timestamp line it can be extremely useful for tracking messages. The **$y** macro is set to the id of the terminal of the sender (if known); some systems like to put this in the Unix "From" line. The **$v** macro is set to be the version number of *sendmail*; this is normally put in timestamps and has been proven extremely useful for debugging. The **$w** macro is set to the name of this host if it can be determined. The **$c** field is set to the "hop count," i.e., the number of times this message has been processed. This can be determined by the **−h** flag on the command line or by counting the timestamps in the message.

The **$r** and **$s** fields are set to the protocol used to communicate with sendmail and the sending hostname; these are not supported in the current version.

Conditionals can be specified using the syntax:

```
$?x text1 $| text2 $.
```

This interpolates *text1* if the macro **$x** is set, and *text2* otherwise. The "else" (**$|**) clause may be omitted.

5.2.2. Special classes

The class **$=w** is set to be the set of all names this host is known by. This can be used to delete local hostnames.

5.2.3. The left hand side

The left hand side of rewriting rules contains a pattern. Normal words are simply matched directly. Metasyntax is introduced using a dollar sign. The metasymbols are:

$*	Match zero or more tokens
$+	Match one or more tokens
$−	Match exactly one token
$=*x*	Match any token in class *x*
$^*x*	Match any token not in class *x*

If any of these match, they are assigned to the symbol **$***n* for replacement on the right hand side, where *n* is the index in the LHS. For example, if the LHS:

    $−:$+

is applied to the input:

    UCBARPA:eric

the rule will match, and the values passed to the RHS will be:

$1	UCBARPA
$2	eric

5.2.4. The right hand side

When the right hand side of a rewriting rule matches, the input is deleted and replaced by the right hand side. Tokens are copied directly from the RHS unless they are begin with a dollar sign. Metasymbols are:

$*n*	Substitute indefinite token *n* from LHS
$>*n*	"Call" ruleset *n*
$#*mailer*	Resolve to *mailer*
$@*host*	Specify *host*
$:*user*	Specify *user*

The **$***n* syntax substitutes the corresponding value from a **$+**, **$−**, **$***, **$=**, or **$^** match on the LHS. It may be used anywhere.

The **$>***n* syntax causes the remainder of the line to be substituted as usual and then passed as the argument to ruleset *n*. The final value of ruleset *n* then becomes the substitution for this rule.

The **$#** syntax should *only* be used in ruleset zero. It causes evaluation of the ruleset to terminate immediately, and signals to sendmail that the address has completely resolved. The complete syntax is:

$#***mailer***$@*host***$:***user*

This specifies the {mailer, host, user} 3-tuple necessary to direct the mailer. If the mailer is local the host part may be omitted. The *mailer* and *host* must be a single word, but the *user* may be multi-part.

A RHS may also be preceeded by a **$@** or a **$:** to control evaluation. A **$@** prefix causes the ruleset to return with the remainder of the RHS as the value. A **$:** prefix causes the rule to terminate immediately, but the ruleset to continue; this can be used to avoid continued application of a rule. The prefix is stripped before continuing.

The **$@** and **$:** prefixes may preceed a **$>** spec; for example:

```
RS+     $:$>7$1
```

matches anything, passes that to ruleset seven, and continues; the **$:** is necessary to avoid an infinite loop.

5.2.5. Semantics of rewriting rule sets

There are five rewriting sets that have specific semantics. These are related as depicted by figure 2.

Ruleset three should turn the address into "canonical form." This form should have the basic syntax:

local-part@host-domain-spec

If no "@" sign is specified, then the host-domain-spec *may* be appended from the sender address (if the C flag is set in the mailer definition corresponding to the *sending* mailer). Ruleset three is applied by sendmail before doing anything with any address.

Ruleset zero is applied after ruleset three to addresses that are going to actually specify recipients. It must resolve to a {*mailer, host, user*} triple. The *mailer* must be defined in the mailer definitions from the configuration file. The *host* is defined into the **$h** macro for use in the argv expansion of the specified mailer.

Rulesets one and two are applied to all sender and recipient addresses respectively. They are applied before any specification in the mailer definition. They must never resolve.

Ruleset four is applied to all addresses in the message. It is typically used to translate internal to external form.

5.2.6. Mailer flags etc.

There are a number of flags that may be associated with each mailer, each identified by a letter of the alphabet. Many of them are assigned semantics internally. These are detailed in Appendix C. Any other flags may be used freely to conditionally assign headers to messages destined for particular mailers.

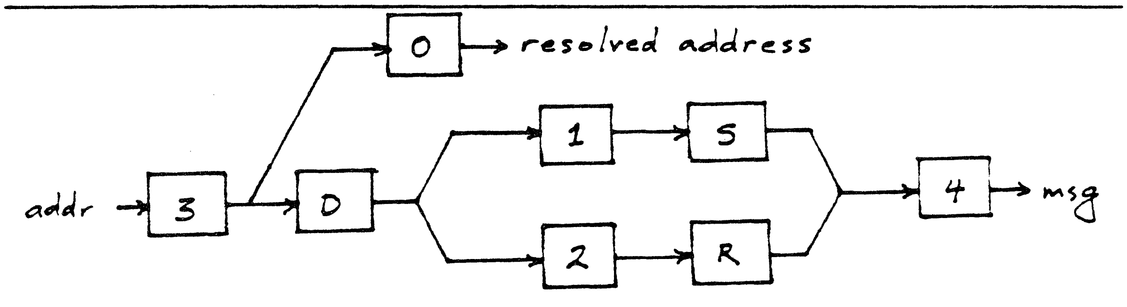

Figure 2 — Rewriting Set Semantics
D—Sender domain addition S—mailer-specific sender rewriting R—mailer-specific recipient rewriting

5.2.7. The "error" mailer

The mailer with the special name "error" can be used to generate a user error. The (optional) host field is a numeric exit status to be returned, and the user field is a message to be printed. For example, the entry:

```
$#error$:Host unknown in this domain
```

on the RHS of a rule will cause the specified error to be generated if the LHS matches. This mailer is only functional in ruleset zero.

5.3. Building a Configuration File From Scratch

Building a configuration table from scratch is an extremely difficult job. Fortunately, it is almost never necessary to do so; nearly every situation that may come up may be resolved by changing an existing table. In any case, it is critical that you understand what it is that you are trying to do and come up with a philosophy for the configuration table. This section is intended to explain what the real purpose of a configuration table is and to give you some ideas for what your philosophy might be.

5.3.1. What you are trying to do

The configuration table has three major purposes. The first and simplest is to set up the environment for *sendmail*. This involves setting the options, defining a few critical macros, etc. Since these are described in other places, we will not go into more detail here.

The second purpose is to rewrite addresses in the message. This should typically be done in two phases. The first phase maps addresses in any format into a canonical form. This should be done in ruleset three. The second phase maps this canonical form into the syntax appropriate for the receiving mailer. *Sendmail* does this in three subphases. Rulesets one and two are applied to all sender and recipient addresses respectively. After this, you may specify per-mailer rulesets for both sender and recipient addresses; this allows mailer-specific customization. Finally, ruleset four is applied to do any default conversion to external form.

The third purpose is to map addresses into the actual set of instructions necessary to get the message delivered. Ruleset zero must resolve to the internal form, which is in turn used as a pointer to a mailer descriptor. The mailer descriptor describes the interface requirements of the mailer.

5.3.2. Philosophy

The particular philosophy you choose will depend heavily on the size and structure of your organization. I will present a few possible philosophies here.

One general point applies to all of these philosophies: it is almost always a mistake to try to do full name resolution. For example, if you are trying to get names of the form "user@host" to the Arpanet, it does not pay to route them to "xyzvax!decvax!ucbvax!c70:user@host" since you then depend on several links not under your control. The best approach to this problem is to simply forward to "xyzvax!user@host" and let xyzvax worry about it from there. In summary, just get the message closer to the destination, rather than determining the full path.

5.3.2.1. Large site, many hosts — minimum information

Berkeley is an example of a large site, i.e., more than two or three hosts. We have decided that the only reasonable philosophy in our environment is to designate one host as the guru for our site. It must be able to resolve any piece of mail it receives. The other sites should have the minimum amount of information they can get away with. In addition, any information they do have should be hints rather than solid information.

For example, a typical site on our local ether network is "monet." Monet has a list of known ethernet hosts; if it receives mail for any of them, it can do direct delivery. If it receives mail for any unknown host, it just passes it directly to "ucbvax," our master host. Ucbvax may determine that the host name is illegal and reject the message, or may be able to do delivery. However, it is important to note that when a new ethernet host is added, the only host that *must* have its tables updated is ucbvax; the others *may* be updated as convenient, but this is not critical.

This picture is slightly muddied due to network connections that are not actually located on ucbvax. For example, our TCP connection is currently on "ucbarpa." However, monet *does not* know about this; the information is hidden totally between ucbvax and ucbarpa. Mail going from monet to a TCP host is transfered via the ethernet from monet to ucbvax, then via the ethernet from ucbvax to ucbarpa, and then is submitted to the Arpanet. Although this involves some extra hops, we feel this is an acceptable tradeoff.

An interesting point is that it would be possible to update monet to send TCP mail directly to ucbarpa if the load got too high; if monet failed to note a host as a TCP host it would go via ucbvax as before, and if monet incorrectly sent a message to ucbarpa it would still be sent by ucbarpa to ucbvax as before. The only problem that can occur is loops, as if ucbarpa thought that ucbvax had the TCP connection and vice versa. For this reason, updates should *always* happen to the master host first.

This philosophy results as much from the need to have a single source for the configuration files (typically built using *m4*(1) or some similar tool) as any logical need. Maintaining more than three separate tables by hand is essentially an impossible job.

5.3.2.2. Small site — complete information

A small site (two or three hosts) may find it more reasonable to have complete information at each host. This would require that each host know exactly where each network connection is, possibly including the names of each host on that network. As long as the site remains small and the the configuration remains relatively static, the update problem will probably not be too great.

5.3.2.3. Single host

This is in some sense the trivial case. The only major issue is trying to insure that you don't have to know too much about your environment. For example, if you have a UUCP connection you might find it useful to know about the names of hosts connected directly to you, but this is really not necessary since this may be determined from the syntax.

5.3.3. Relevant issues

The canonical form you use should almost certainly be as specified in the Arpanet protocols RFC819 and RFC822. Copies of these RFC's are included on the *sendmail* tape as *doc/rfc819.lpr* and *doc/rfc822.lpr.*

RFC822 describes the format of the mail message itself. *Sendmail* follows this RFC closely, to the extent that many of the standards described in this document can not be changed without changing the code. In particular, the following characters have special interpretations:

< > ( ) " \

Any attempt to use these characters for other than their RFC822 purpose in addresses is probably doomed to disaster.

RFC819 describes the specifics of the domain-based addressing. This is touched on in RFC822 as well. Essentially each host is given a name which is a right-to-left dot qualified pseudo-path from a distinguished root. The elements of the path need not be physical hosts; the domain is logical rather than physical. For example, at Berkeley one legal host is "a.cc.berkeley.arpa"; reading from right to left, "arpa" is a top level domain (related to, but not limited to, the physical Arpanet), "berkeley" is both an Arpanet host and a logical domain which is actually interpreted by a host called ucbvax (which is actually just the "major" host for this domain), "cc" represents the Computer Center, (in this case a strictly logical entity), and "a" is a host in the Computer Center; this particular host happens to be connected via berknet, but other hosts might be connected via one of two ethernets or some other network.

Beware when reading RFC819 that there are a number of errors in it.

5.3.4. How to proceed

Once you have decided on a philosophy, it is worth examining the available configuration tables to decide if any of them are close enough to steal major parts of. Even under the worst of conditions, there is a fair amount of boiler plate that can be collected safely.

The next step is to build ruleset three. This will be the hardest part of the job. Beware of doing too much to the address in this ruleset, since anything you do will reflect through to the message. In particular, stripping of local domains is best deferred, since this can leave you with addresses with no domain spec at all. Since *sendmail* likes to append the sending domain to addresses with no domain, this can change the semantics of addresses. Also try to avoid fully qualifying domains in this ruleset. Although technically legal, this can lead to unpleasantly and unnecessarily long addresses reflected into messages. The Berkeley configuration files define ruleset nine to qualify domain names and strip local domains. This is called from ruleset zero to get all addresses into a cleaner form.

Once you have ruleset three finished, the other rulesets should be relatively trivial. If you need hints, examine the supplied configuration tables.

5.3.5. Testing the rewriting rules − the −bt flag

When you build a configuration table, you can do a certain amount of testing using the "test mode" of *sendmail*. For example, you could invoke *sendmail* as:

```
sendmail −bt −Ctest.cf
```

which would read the configuration file "test.cf" and enter test mode. In this mode, you enter lines of the form:

```
rwset address
```

where *rwset* is the rewriting set you want to use and *address* is an address to apply the set to. Test mode shows you the steps it takes as it proceeds, finally showing you the address it ends up with. You may use a comma separated list of rwsets for sequential application of rules to an input; ruleset three is always applied first. For example:

```
1,21,4 monet:bollard
```

first applies ruleset three to the input "monet:bollard." Ruleset one is then applied to the output of ruleset three, followed similarly by rulesets twenty-one and four.

If you need more detail, you can also use the "−d21" flag to turn on more debugging. For example,

```
sendmail −bt −d21.99
```

turns on an incredible amount of information; a single word address is probably going to print out several pages worth of information.

5.3.6. Building mailer descriptions

To add an outgoing mailer to your mail system, you will have to define the characteristics of the mailer.

Each mailer must have an internal name. This can be arbitrary, except that the names "local" and "prog" must be defined.

The pathname of the mailer must be given in the P field. If this mailer should be accessed via an IPC connection, use the string "[IPC]" instead.

The F field defines the mailer flags. You should specify an "f" or "r" flag to pass the name of the sender as a **−f** or **−r** flag respectively. These flags are only passed if they were passed to *sendmail*, so that mailers that give errors under some circumstances can be placated. If the mailer is not picky you can just specify "**−f $g**" in the argv template. If the mailer must be called as **root** the "S" flag should be given; this will not reset the userid before calling the mailer[3]. If this mailer is local (i.e., will perform final delivery rather than another network hop) the "l" flag should be given. Quote characters (backslashes and " marks) can be stripped from addresses if the "s" flag is specified; if this is not given they are passed through. If the mailer is capable of sending to more than one user on the same host in a single transaction the "m" flag should be stated. If this flag is on, then the argv template containing **$u** will be repeated for each unique user on a given host. The "e" flag will mark the mailer as being "expensive," which will cause *sendmail* to defer connection until a queue run[4].

An unusual case is the "C" flag. This flag applies to the mailer that the message is received from, rather than the mailer being sent to; if set, the domain spec of the sender (i.e., the "@host.domain" part) is saved and is appended to any addresses in the message that do not already contain a domain spec. For example, a message of the form:

```
From: eric@ucbarpa
To: wnj@monet, mckusick
```

will be modified to:

```
From: eric@ucbarpa
To: wnj@monet, mckusick@ucbarpa
```

if and only if the "C" flag is defined in the mailer corresponding to "eric@ucbarpa."

Other flags are described in Appendix C.

The S and R fields in the mailer description are per-mailer rewriting sets to be applied to sender and recipient addresses respectively. These are applied after the sending domain is appended and the general rewriting sets (numbers one and two) are applied, but before the output rewrite (ruleset four) is applied. A typical use is to append the current domain to addresses that do not already have a domain. For example, a header of the form:

```
From: eric
```

might be changed to be:

```
From: eric@ucbarpa
```

or

[3] *Sendmail* must be running setuid to root for this to work.

[4] The "c" configuration option must be given for this to be effective.

From: ucbvax!eric

depending on the domain it is being shipped into. These sets can also be used to do special purpose output rewriting in cooperation with ruleset four.

The E field defines the string to use as an end-of-line indication. A string containing only newline is the default. The usual backslash escapes (\r, \n, \f, \b) may be used.

Finally, an argv template is given as the E field. It may have embedded spaces. If there is no argv with a **$u** macro in it, *sendmail* will speak SMTP to the mailer. If the pathname for this mailer is "[IPC]," the argv should be

```
IPC $h [ port ]
```

where *port* is the optional port number to connect to.

For example, the specifications:

```
Mlocal, P=/bin/mail, F=rlsm  S=10, R=20, A=mail −d $u
Mether,P=[IPC],      F=meC,S=11, R=21, A=IPC $h, M=100000
```

specifies a mailer to do local delivery and a mailer for ethernet delivery. The first is called "local," is located in the file "/bin/mail," takes a picky **−r** flag, does local delivery, quotes should be stripped from addresses, and multiple users can be delivered at once; ruleset ten should be applied to sender addresses in the message and ruleset twenty should be applied to recipient addresses; the argv to send to a message will be the word "mail," the word "−d," and words containing the name of the receiving user. If a **−r** flag is inserted it will be between the words "mail" and "−d." The second mailer is called "ether," it should be connected to via an IPC connection, it can handle multiple users at once, connections should be deferred, and any domain from the sender address should be appended to any receiver name without a domain; sender addresses should be processed by ruleset eleven and recipient addresses by ruleset twenty-one. There is a 100,000 byte limit on messages passed through this mailer.

APPENDIX A

COMMAND LINE FLAGS

Arguments must be presented with flags before addresses. The flags are:

−f *addr*	The sender's machine address is *addr*. This flag is ignored unless the real user is listed as a "trusted user" or if *addr* contains an exclamation point (because of certain restrictions in UUCP).
−r *addr*	An obsolete form of −f.
−h *cnt*	Sets the "hop count" to *cnt*. This represents the number of times this message has been processed by *sendmail* (to the extent that it is supported by the underlying networks). *Cnt* is incremented during processing, and if it reaches MAXHOP (currently 30) *sendmail* throws away the message with an error.
−F*name*	Sets the full name of this user to *name*.
−n	Don't do aliasing or forwarding.
−t	Read the header for "To:", "Cc:", and "Bcc:" lines, and send to everyone listed in those lists. The "Bcc:" line will be deleted before sending. Any addresses in the argument vector will be deleted from the send list.
−b*x*	Set operation mode to *x*. Operation modes are:

m	Deliver mail (default)
a	Run in arpanet mode (see below)
s	Speak SMTP on input side
d	Run as a daemon
t	Run in test mode
v	Just verify addresses, don't collect or deliver
i	Initialize the alias database
p	Print the mail queue
z	Freeze the configuration file

The special processing for the ARPANET includes reading the "From:" line from the header to find the sender, printing ARPANET style messages (preceded by three digit reply codes for compatibility with the FTP protocol [Neigus73, Postel74, Postel77]), and ending lines of error messages with <CRLF>.

−q*time*	Try to process the queued up mail. If the time is given, a sendmail will run through the queue at the specified interval to deliver queued mail; otherwise, it only runs once.
−C*file*	Use a different configuration file.
−d*level*	Set debugging level.
−o*x value*	Set option *x* to the specified *value*. These options are described in Appendix B.

There are a number of options that may be specified as primitive flags (provided for compatibility with *delivermail*). These are the e, i, m, and v options. Also, the f option may be specified as the −s flag.

APPENDIX B

CONFIGURATION OPTIONS

The following options may be set using the **−o** flag on the command line or the **O** line in the configuration file:

A*file*	Use the named *file* as the alias file. If no file is specified, use *aliases* in the current directory.
a	If set, wait for an "@:@" entry to exist in the alias database before starting up. If it does not appear in five minutes, rebuild the database.
c	If an outgoing mailer is marked as being expensive, don't connect immediately. This requires that queueing be compiled in, since it will depend on a queue run process to actually send the mail.
d*x*	Deliver in mode *x*. Legal modes are: i Deliver interactively (synchronously) b Deliver in background (asynchronously) q Just queue the message (deliver during queue run)
D	If set, rebuild the alias database if necessary and possible. If this option is not set, *sendmail* will never rebuild the alias database unless explicitly requested using **−bi**.
e*x*	Dispose of errors using mode *x*. The values for *x* are: p Print error messages (default) q No messages, just give exit status m Mail back errors w Write back errors (mail if user not logged in) e Mail back errors and give zero exit stat always
F*n*	The temporary file mode, in octal. 644 and 600 are good choices.
f	Save Unix-style "From" lines at the front of headers. Normally they are assumed redundant and discarded.
g*n*	Set the default group id for mailers to run in to *n*.
H*file*	Specify the help file for SMTP.
i	Ignore dots in incoming messages.
L*n*	Set the default log level to *n*.
M*x value*	Set the macro *x* to *value*. This is intended only for use from the command line.
m	Send to me too, even if I am in an alias expansion.
o	Assume that the headers may be in old format, i.e., spaces delimit names. This actually turns on an adaptive algorithm: if any recipient address contains a comma, parenthesis, or angle bracket, it will be assumed that commas already exist. If this flag is not on, only commas delimit names. Headers are always output with commas between the names.
Q*dir*	Use the named *dir* as the queue directory.
r*time*	Timeout reads after *time* interval.

S*file*	Log statistics in the named *file*.
s	Be super-safe when running things, i.e., always instantiate the queue file, even if you are going to attempt immediate delivery. *Sendmail* always instantiates the queue file before returning control the the client under any circumstances.
T*time*	Set the queue timeout to *time*. After this interval, messages that have not been successfully sent will be returned to the sender.
t*S,D*	Set the local timezone name to *S* for standard time and *D* for daylight time; this is only used under version six.
u*n*	Set the default userid for mailers to *n*. Mailers without the *S* flag in the mailer definition will run as this user.
v	Run in verbose mode.

APPENDIX C

MAILER FLAGS

The following flags may be set in the mailer description.

f	The mailer wants a **−f** *from* flag, but only if this is a network forward operation (i.e., the mailer will give an error if the executing user does not have special permissions).
r	Same as f, but sends a **−r** flag.
S	Don't reset the userid before calling the mailer. This would be used in a secure environment where *sendmail* ran as root. This could be used to avoid forged addresses. This flag is suppressed if given from an "unsafe" environment (e.g, a user's mail.cf file).
n	Do not insert a UNIX-style "From" line on the front of the message.
l	This mailer is local (i.e., final delivery will be performed).
s	Strip quote characters off of the address before calling the mailer.
m	This mailer can send to multiple users on the same host in one transaction. When a **$u** macro occurs in the *argv* part of the mailer definition, that field will be repeated as necessary for all qualifying users.
F	This mailer wants a "From:" header line.
D	This mailer wants a "Date:" header line.
M	This mailer wants a "Message-Id:" header line.
x	This mailer wants a "Full-Name:" header line.
P	This mailer wants a "Return-Path:" line.
u	Upper case should be preserved in user names for this mailer.
h	Upper case should be preserved in host names for this mailer.
A	This is an Arpanet-compatible mailer, and all appropriate modes should be set.
U	This mailer wants Unix-style "From" lines with the ugly UUCP-style "remote from <host>" on the end.
e	This mailer is expensive to connect to, so try to avoid connecting normally; any necessary connection will occur during a queue run.
X	This mailer want to use the hidden dot algorithm as specified in RFC821; basically, any line beginning with a dot will have an extra dot prepended (to be stripped at the other end). This insures that lines in the message containing a dot will not terminate the message prematurely.
L	Limit the line lengths as specified in RFC821.
P	Use the return-path in the SMTP "MAIL FROM:" command rather than just the return address; although this is required in RFC821, many hosts do not process return paths properly.
I	This mailer will be speaking SMTP to another *sendmail* — as such it can use special protocol features. This option is not required (i.e., if this option is omitted the transmission will still operate successfully, although perhaps not as efficiently as possible).
C	If mail is *received* from a mailer with this flag set, any addresses in the header that do not have an at sign ("@") after being rewritten by ruleset three will have the "@domain" clause from the sender tacked on. This allows mail with headers of the form:

```
From: usera@hosta
To: userb@hostb, userc
```

to be rewritten as:

```
From: usera@hosta
To: userb@hostb, userc@hosta
```

automatically.

APPENDIX D

OTHER CONFIGURATION

There are some configuration changes that can be made by recompiling *sendmail.* These are located in three places:

md/config.m4	These contain operating-system dependent descriptions. They are interpolated into the Makefiles in the *src* and *aux* directories. This includes information about what version of UNIX you are running, what libraries you have to include, etc.
src/conf.h	Configuration parameters that may be tweaked by the installer are included in conf.h.
src/conf.c	Some special routines and a few variables may be defined in conf.c. For the most part these are selected from the settings in conf.h.

Parameters in md/config.m4

The following compilation flags may be defined in the *m4CONFIG* macro in *md/config.m4* to define the environment in which you are operating.

V6	If set, this will compile a version 6 system, with 8-bit user id's, single character tty id's, etc.
VMUNIX	If set, you will be assumed to have a Berkeley 4BSD or 4.1BSD, including the *vfork*(2) system call, special types defined in <sys/types.h> (e.g, u_char), etc.

If none of these flags are set, a version 7 system is assumed.

You will also have to specify what libraries to link with *sendmail* in the *m4LIBS* macro. Most notably, you will have to include if you are running a 4.1BSD system.

Parameters in src/conf.h

Parameters and compilation options are defined in conf.h. Most of these need not normally be tweaked; common parameters are all in sendmail.cf. However, the sizes of certain primitive vectors, etc., are included in this file. The numbers following the parameters are their default value.

MAXLINE [256]	The maximum line length of any input line. If message lines exceed this length they will still be processed correctly; however, header lines, configuration file lines, alias lines, etc., must fit within this limit.
MAXNAME [128]	The maximum length of any name, such as a host or a user name.
MAXFIELD [2500]	The maximum total length of any header field, including continuation lines.
MAXPV [40]	The maximum number of parameters to any mailer. This limits the number of recipients that may be passed in one transaction.
MAXHOP [30]	When a message has been processed more than this number of times, sendmail rejects the message on the assumption that there has been an aliasing loop. This can be determined from the **−h** flag or by counting the number of trace fields (i.e, "Received:" lines) in the message header.
MAXATOM [100]	The maximum number of atoms (tokens) in a single address. For example, the address "eric@Berkeley" is three atoms.

MAXMAILERS [25]
: The maximum number of mailers that may be defined in the configuration file.

MAXRWSETS [30]
: The maximum number of rewriting sets that may be defined.

MAXPRIORITIES [25]
: The maximum number of values for the "Precedence:" field that may be defined (using the **P** line in sendmail.cf).

MAXTRUST [30] The maximum number of trusted users that may be defined (using the T line in sendmail.cf).

A number of other compilation options exist. These specify whether or not specific code should be compiled in.

DBM	If set, the "DBM" package in UNIX is used (see DBM(3X) in [UNIX80]). If not set, a much less efficient algorithm for processing aliases is used.
DEBUG	If set, debugging information is compiled in. To actually get the debugging output, the **−d** flag must be used.
LOG	If set, the *syslog* routine in use at some sites is used. This makes an informational log record for each message processed, and makes a higher priority log record for internal system errors.
QUEUE	This flag should be set to compile in the queueing code. If this is not set, mailers must accept the mail immediately or it will be returned to the sender.
SMTP	If set, the code to handle user and server SMTP will be compiled in. This is only necessary if your machine has some mailer that speaks SMTP.
DAEMON	If set, code to run a daemon is compiled in. This code is for 4.2BSD if the NVMUNIX flag is specified; otherwise, 4.1a BSD code is used. Beware however that there are bugs in the 4.1a code that make it impossible for **sendmail** to work correctly under heavy load.
UGLYUUCP	If you have a UUCP host adjacent to you which is not running a reasonable version of *rmail*, you will have to set this flag to include the "remote from sysname" info on the from line. Otherwise, UUCP gets confused about where the mail came from.
NOTUNIX	If you are using a non-UNIX mail format, you can set this flag to turn off special processing of UNIX-style "From " lines.

Configuration in src/conf.c

Not all header semantics are defined in the configuration file. Header lines that should only be included by certain mailers (as well as other more obscure semantics) must be specified in the *HdrInfo* table in *conf.c*. This table contains the header name (which should be in all lower case) and a set of header control flags (described below), The flags are:

H_ACHECK	Normally when the check is made to see if a header line is compatible with a mailer, *sendmail* will not delete an existing line. If this flag is set, *sendmail* will delete even existing header lines. That is, if this bit is set and the mailer does not have flag bits set that intersect with the required mailer flags in the header definition in sendmail.cf, the header line is *always* deleted.
H_EOH	If this header field is set, treat it like a blank line, i.e., it will signal the end of the header and the beginning of the message text.
H_FORCE	Add this header entry even if one existed in the message before. If a header entry does not have this bit set, *sendmail* will not add another header line if a header line of this name already existed. This would normally be used to stamp the message by everyone who handled it.

H_TRACE	If set, this is a timestamp (trace) field. If the number of trace fields in a message exceeds a preset amount the message is returned on the assumption that it has an aliasing loop.
H_RCPT	If set, this field contains recipient addresses. This is used by the **−t** flag to determine who to send to when it is collecting recipients from the message.
H_FROM	This flag indicates that this field specifies a sender. The order of these fields in the *HdrInfo* table specifies *sendmail's* preference for which field to return error messages to.

Let's look at a sample *HdrInfo* specification:

```
struct hdrinfo          HdrInfo[] =
{
        /* originator fields, most to least significant */
    "resent-sender",    H_FROM,
    "resent-from",      H_FROM,
    "sender",           H_FROM,
    "from",             H_FROM,
    "full-name",        H_ACHECK,
        /* destination fields */
    "to",               H_RCPT,
    "resent-to",        H_RCPT,
    "cc",               H_RCPT,
        /* message identification and control */
    "message",          H_EOH,
    "text",             H_EOH,
        /* trace fields */
    "received",         H_TRACE|H_FORCE,

    NULL,               0,
};
```

This structure indicates that the "To:", "Resent-To:", and "Cc:" fields all specify recipient addresses. Any "Full-Name:" field will be deleted unless the required mailer flag (indicated in the configuration file) is specified. The "Message:" and "Text:" fields will terminate the header; these are specified in new protocols [NBS80] or used by random dissenters around the network world. The "Received:" field will always be added, and can be used to trace messages.

There are a number of important points here. First, header fields are not added automatically just because they are in the *HdrInfo* structure; they must be specified in the configuration file in order to be added to the message. Any header fields mentioned in the configuration file but not mentioned in the *HdrInfo* structure have default processing performed; that is, they are added unless they were in the message already. Second, the *HdrInfo* structure only specifies cliched processing; certain headers are processed specially by ad hoc code regardless of the status specified in *HdrInfo*. For example, the "Sender:" and "From:" fields are always scanned on ARPANET mail to determine the sender; this is used to perform the "return to sender" function. The "From:" and "Full-Name:" fields are used to determine the full name of the sender if possible; this is stored in the macro **$x** and used in a number of ways.

The file *conf.c* also contains the specification of ARPANET reply codes. There are four classifications these fall into:

```
char Arpa_Info[] =     "050";  /* arbitrary info */
char Arpa_TSyserr[] =  "455";  /* some (transient) system error */
char Arpa_PSyserr[] =  "554";  /* some (transient) system error */
char Arpa_Usrerr[] =   "554";  /* some (fatal) user error */
```

The class *Arpa_Info* is for any information that is not required by the protocol, such as forwarding information. *Arpa_TSyserr* and *Arpa_PSyserr* is printed by the *syserr* routine. TSyserr is printed out for transient errors, whereas PSyserr is printed for permanent errors: the distinction is made based on the value of *errno*. Finally, *Arpa_Usrerr* is the result of a user error and is generated by the *usrerr* routine; these are generated when the user has specified something wrong, and hence the error is permanent, i.e., it will not work simply by resubmitting the request.

If it is necessary to restrict mail through a relay, the *checkcompat* routine can be modified. This routine is called for every recipient address. It can return **TRUE** to indicate that the address is acceptable and mail processing will continue, or it can return **FALSE** to reject the recipient. If it returns false, it is up to *checkcompat* to print an error message (using *usrerr)* saying why the message is rejected. For example, *checkcompat* could read:

```
bool
checkcompat(to)
    register ADDRESS *to;
{
    if (MsgSize > 50000 && to->q_mailer != LocalMailer)
    {
        usrerr("Message too large for non-local delivery");
        NoReturn = TRUE;
        return (FALSE);
    }
    return (TRUE);
}
```

This would reject messages greater than 50000 bytes unless they were local. The *NoReturn* flag can be sent to supress the return of the actual body of the message in the error return. The actual use of this routine is highly dependent on the implementation, and use should be limited.

APPENDIX E

SUMMARY OF SUPPORT FILES

This is a summary of the support files that *sendmail* creates or generates.

/usr/lib/sendmail	The binary of *sendmail.*
/usr/bin/newaliases	A link to /usr/lib/sendmail; causes the alias database to be rebuilt. Running this program is completely equivalent to giving *sendmail* the **−bi** flag.
/usr/bin/mailq	Prints a listing of the mail queue. This program is equivalent to using the **−bp** flag to *sendmail.*
/usr/lib/sendmail.cf	The configuration file, in textual form.
/usr/lib/sendmail.fc	The configuration file represented as a memory image.
/usr/lib/sendmail.hf	The SMTP help file.
/usr/lib/sendmail.st	A statistics file; need not be present.
/usr/lib/aliases	The textual version of the alias file.
/usr/lib/aliases.{pag,dir}	The alias file in *dbm*(3) format.
/etc/syslog	The program to do logging.
/etc/syslog.conf	The configuration file for syslog.
/etc/syslog.pid	Contains the process id of the currently running syslog.
/usr/spool/mqueue	The directory in which the mail queue and temporary files reside.
/usr/spool/mqueue/qf*	Control (queue) files for messages.
/usr/spool/mqueue/df*	Data files.
/usr/spool/mqueue/lf*	Lock files
/usr/spool/mqueue/tf*	Temporary versions of the qf files, used during queue file rebuild.
/usr/spool/mqueue/nf*	A file used when creating a unique id.
/usr/spool/mqueue/xf*	A transcript of the current session.

4.2BSD Line Printer Spooler Manual

Revised July 27, 1983

Ralph Campbell

Computer Systems Research Group
Computer Science Division
Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA 94720

(415) 642-7780

ABSTRACT

This document describes the structure and installation procedure for the line printer spooling system developed for the 4.2BSD version of the UNIX* operating system.

1. Overview

The line printer system supports:

- multiple printers,
- multiple spooling queues,
- both local and remote printers, and
- printers attached via serial lines which require line initialization such as the baud rate.

Raster output devices such as a Varian or Versatec, and laser printers such as an Imagen, are also supported by the line printer system.

The line printer system consists mainly of the following files and commands:

/etc/printcap	printer configuration and capability data base
/usr/lib/lpd	line printer daemon, does all the real work
/usr/ucb/lpr	program to enter a job in a printer queue
/usr/ucb/lpq	spooling queue examination program
/usr/ucb/lprm	program to delete jobs from a queue
/etc/lpc	program to administer printers and spooling queues
/dev/printer	socket on which lpd listens

The file /etc/printcap is a master data base describing line printers directly attached to a machine and, also, printers accessible across a network. The manual page entry *printcap*(5) provides the ultimate definition of the format of this data base, as well as indicating default values for important items such as the directory in which spooling is performed. This document highlights the important information which may be placed *printcap*.

* UNIX is a trademark of Bell Laboratories.

2. Commands

2.1. lpd — line printer dameon

The program *lpd*(8), usually invoked at boot time from the /etc/rc file, acts as a master server for coordinating and controlling the spooling queues configured in the printcap file. When *lpd* is started it makes a single pass through the *printcap* database restarting any printers which have jobs. In normal operation *lpd* listens for service requests on multiple sockets, one in the UNIX domain (named "/dev/printer") for local requests, and one in the Internet domain (under the "printer" service specification) for requests for printer access from off machine; see *socket*(2) and *services*(5) for more information on sockets and service specifications, respectively. *Lpd* spawns a copy of itself to process the request; the master daemon continues to listen for new requests.

Clients communicate with *lpd* using a simple transaction oriented protocol. Authentication of remote clients is done based on the "privilege port" scheme employed by *rshd*(8C) and *rcmd*(3X). The following table shows the requests understood by *lpd*. In each request the first byte indicates the "meaning" of the request, followed by the name of the printer to which it should be applied. Additional qualifiers may follow, depending on the request.

Request	Interpretation
^Aprinter\n	check the queue for jobs and print any found
^Bprinter\n	receive and queue a job from another machine
^Cprinter [users ...] [jobs ...]\n	return short list of current queue state
^Dprinter [users ...] [jobs ...]\n	return long list of current queue state
^Eprinter person [users ...] [jobs ...]\n	remove jobs from a queue

The *lpr*(1) command is used by users to enter a print job in a local queue and to notify the local *lpd* that there are new jobs in the spooling area. *Lpd* either schedules the job to be printed locally, or in the case of remote printing, attempts to forward the job to the appropriate machine. If the printer cannot be opened or the destination machine is unreachable, the job will remain queued until it is possible to complete the work.

2.2. lpq — show line printer queue

The *lpq*(1) program works recursively backwards displaying the queue of the machine with the printer and then the queue(s) of the machine(s) that lead to it. *Lpq* has two forms of output: in the default, short, format it gives a single line of output per queued job; in the long format it shows the list of files, and their sizes, which comprise a job.

2.3. lprm — remove jobs from a queue

The *lprm*(1) command deletes jobs from a spooling queue. If necessary, *lprm* will first kill off a running daemon which is servicing the queue, restarting it after the required files are removed. When removing jobs destined for a remote printer, *lprm* acts similarly to *lpq* except it first checks locally for jobs to remove and then tries to remove files in queues off-machine.

2.4. lpc — line printer control program

The *lpc*(8) program is used by the system administrator to control the operation of the line printer system. For each line printer configured in /etc/printcap, *lpc* may be used to:

- disable or enable a printer,
- disable or enable a printer's spooling queue,
- rearrange the order of jobs in a spooling queue,
- find the status of printers, and their associated spooling queues and printer dameons.

3. Access control

The printer system maintains protected spooling areas so that users cannot circumvent printer accounting or remove files other than their own. The strategy used to maintain protected spooling areas is as follows:

- The spooling area is writable only by a *daemon* user and *spooling* group.
- The *lpr* program runs setuid *root* and setgid *spooling*. The *root* access is used to read any file required, verifying accessibility with an *access*(2) call. The group ID is used in setting up proper ownership of files in the spooling area for *lprm*.
- Control files in a spooling area are made with *daemon* ownership and group ownership *spooling*. Their mode is 0660. This insures control files are not modified by a user and that no user can remove files except through *lprm*.
- The spooling programs, *lpd*, *lpq*, and *lprm* run setuid *root* and setgid *spooling* to access spool files and printers.
- The printer server, *lpd*, uses the same verification procedures as *rshd*(8C) in authenticating remote clients. The host on which a client resides must be present in the file /etc/hosts.equiv, used to create clusters of machines under a single administration.

In practice, none of *lpd*, *lpq*, or *lprm* would have to run as user *root* if remote spooling were not supported. In previous incarnations of the printer system *lpd* ran setuid *daemon*, setgid *spooling*, and *lpq* and *lprm* ran setgid *spooling*.

4. Setting up

The 4.2BSD release comes with the necessary programs installed and with the default line printer queue created. If the system must be modified, the makefile in the directory /usr/src/usr.lib/lpr should be used in recompiling and reinstalling the necessary programs.

The real work in setting up is to create the *printcap* file and any printer filters for printers not supported in the distribution system.

4.1. Creating a printcap file

The *printcap* database contains one or more entries per printer. A printer should have a separate spooling directory; otherwise, jobs will be printed on different printers depending on which printer daemon starts first. This section describes how to create entries for printers which do not conform to the default printer description (an LP-11 style interface to a standard, band printer).

4.1.1. Printers on serial lines

When a printer is connected via a serial communication line it must have the proper baud rate and terminal modes set. The following example is for a DecWriter III printer connected locally via a 1200 baud serial line.

```
lp|LA-180 DecWriter III:\
	:lp=/dev/lp:br#1200:fs#06320:\
	:tr=\f:of=/usr/lib/lpf:lf=/usr/adm/lpd-errs:
```

The **lp** entry specifies the file name to open for output. In this case it could be left out since "/dev/lp" is the default. The **br** entry sets the baud rate for the tty line and the **fs** entry sets CRMOD, no parity, and XTABS (see *tty*(4)). The **tr** entry indicates a form-feed should be printed when the queue empties so the paper can be torn off without turning the printer off-line and pressing form feed. The **of** entry specifies the filter program *lpf* should be used for printing the files; more will be said about filters later. The last entry causes errors to be written to the file "/usr/adm/lpd-errs" instead of the console.

4.1.2. Remote printers

Printers which reside on remote hosts should have an empty **lp** entry. For example, the following printcap entry would send output to the printer named "lp" on the machine "ucbvax".

```
lp|default line printer:\
        :lp=:rm=ucbvax:rp=lp:sd=/usr/spool/vaxlpd:
```

The **rm** entry is the name of the remote machine to connect to; this name must appear in the /etc/hosts database, see *hosts*(5). The **rp** capability indicates the name of the printer on the remote machine is "lp"; in this case it could be left out since this is the default value. The **sd** entry specifies "/usr/spool/vaxlpd" as the spooling directory instead of the default value of "/usr/spool/lpd".

4.2. Output filters

Filters are used to handle device dependencies and to perform accounting functions. The output filter **of** is used to filter text data to the printer device when accounting is not used or when all text data must be passed through a filter. It is not intended to perform accounting since it is started only once, all text files are filtered through it, and no provision is made for passing owners' login name, identifying the begining and ending of jobs, etc. The other filters (if specified) are started for each file printed and perform accounting if there is an **af** entry. If entries for both **of** and one of the other filters are specified, the output filter is used only to print the banner page; it is then stopped to allow other filters access to the printer. An example of a printer which requires output filters is the Benson-Varian.

```
va|varian|Benson-Varian:\
        :lp=/dev/va0:sd=/usr/spool/vad:of=/usr/lib/vpf:\
        :tf=/usr/lib/rvcat:mx#2000:pl#58:tr=\f:
```

The **tf** entry specifies "/usr/lib/rvcat" as the filter to be used in printing *troff*(1) output. This filter is needed to set the device into print mode for text, and plot mode for printing *troff* files and raster images (see *va*(4V)). Note that the page length is set to 58 lines by the **pl** entry for 8.5" by 11" fan-fold paper. To enable accounting, the varian entry would be augmented with an **af** filter as shown below.

```
va|varian|Benson-Varian:\
        :lp=/dev/va0:sd=/usr/spool/vad:of=/usr/lib/vpf:\
        :if=/usr/lib/vpf:tf=/usr/lib/rvcat:af=/usr/adm/vaacct:\
        :mx#2000:pl#58:tr=\f:
```

5. Output filter specifications

The filters supplied with 4.2BSD handle printing and accounting for most common line printers, the Benson-Varian, the wide (36") and narrow (11") Versatec printer/plotters. For other devices or accounting methods, it may be necessary to create a new filter.

Filters are spawned by *lpd* with their standard input the data to be printed, and standard output the printer. The standard error is attached to the **lf** file for logging errors. A filter must return a 0 exit code if there were no errors, 1 if the job should be reprinted, and 2 if the job should be thrown away. When *lprm* sends a kill signal to the *lpd* process controlling printing, it sends a SIGINT signal to all filters and descendents of filters. This signal can be trapped by filters which need to perform cleanup operations such as deleting temporary files.

Arguments passed to a filter depend on its type. The **of** filter is called with the following arguments.

ofiler −**w**width −**l**length

The *width* and *length* values come from the **pw** and **pl** entries in the printcap database. The **if** filter is passed the following parameters.

filter [**−c**] **−w**width **−l**length **−i**indent **−n** login **−h** host accounting_file

The **−c** flag is optional, and only supplied when control characters are to be passed uninterpreted to the printer (when the **−l** option of *lpr* is used to print the file). The **−w** and **−l** parameters are the same as for the **of** filter. The **−n** and **−h** parameters specify the login name and host name of the job owner. The last argument is the name of the accounting file from *printcap*.

All other filters are called with the following arguments:

filter **−x**width **−y**length **−n** login **−h** host accounting_file

The **−x** and **−y** options specify the horizontal and vertical page size in pixels (from the **px** and **py** entries in the printcap file). The rest of the arguments are the same as for the **if** filter.

6. Line printer Administration

The *lpc* program provides local control over line printer activity. The major commands and their intended use will be described. The command format and remaining commands are described in *lpc*(8).

abort and **start**

Abort terminates an active spooling daemon on the local host immediately and then disables printing (preventing new daemons from being started by *lpr*). This is normally used to forcibily restart a hung line printer daemon (i.e., *lpq* reports that there is a daemon present but nothing is happening). It does not remove any jobs from the queue (use the *lprm* command instead). *Start* enables printing and requests *lpd* to start printing jobs.

enable and **disable**

Enable and *disable* allow spooling in the local queue to be turned on/off. This will allow/prevent *lpr* from putting new jobs in the spool queue. It is frequently convenient to turn spooling off while testing new line printer filters since the *root* user can still use *lpr* to put jobs in the queue but no one else can. The other main use is to prevent users from putting jobs in the queue when the printer is expected to be unavailable for a long time.

restart

Restart allows ordinary users to restart printer daemons when *lpq* reports that there is no daemon present.

stop

Stop is used to halt a spooling daemon after the current job completes; this also disables printing. This is a clean way to shutdown a printer in order to perform maintenence, etc. Note that users can still enter jobs in a spool queue while a printer is *stopped.*

topq

Topq places jobs at the top of a printer queue. This can be used to reorder high priority jobs since *lpr* only only provides first-come-first-serve ordering of jobs.

7. Troubleshooting

There are a number of messages which may be generated by the the line printer system. This section categorizes the most common and explains the cause for their generation. Where the message indicates a failure, directions are given to remedy the problem.

In the examples below, the name *printer* is the name of the printer. This would be one of the names from the *printcap* database.

7.1. LPR

lpr: ***printer*****: unknown printer**

The *printer* was not found in the *printcap* database. Usually this is a typing mistake; however, it may indicate a missing or incorrect entry in the /etc/printcap file.

lpr: ***printer*****: jobs queued, but cannot start daemon.**

The connection to *lpd* on the local machine failed. This usually means the printer server started at boot time has died or is hung. Check the local socket /dev/printer to be sure it still exists (if it does not exist, there is no *lpd* process running). Use

```
% ps ax | fgrep lpd
```

to get a list of process identifiers of running lpd's. The *lpd* to kill is the one which is not listed in any of the "lock" files (the lock file is contained in the spool directory of each printer). Kill the master daemon using the following command.

```
% kill pid
```

Then remove /dev/printer and restart the daemon (and printer) with the following commands.

```
% rm /dev/printer
% /usr/lib/lpd
```

Another possibility is that the *lpr* program is not setuid *root*, setgid *spooling*. This can be checked with

```
% ls −lg /usr/ucb/lpr
```

lpr: ***printer*****: printer queue is disabled**

This means the queue was turned off with

```
% lpc disable printer
```

to prevent *lpr* from putting files in the queue. This is normally done by the system manager when a printer is going to be down for a long time. The printer can be turned back on by a super-user with *lpc*.

7.2. LPQ

waiting for ***printer*** **to become ready (offline ?)**

The printer device could not be opened by the daemon. This can happen for a number of reasons, the most common being that the printer is turned off-line. This message can also be generated if the printer is out of paper, the paper is jammed, etc. The actual reason is dependent on the meaning of error codes returned by system device driver. Not all printers supply sufficient information to distinguish when a printer is off-line or having trouble (e.g. a printer connected through a serial line). Another possible cause of this message is some other process, such as an output filter, has an exclusive open on the device. Your only recourse here is to kill off the offending program(s) and restart the printer with *lpc*.

printer **is ready and printing**

The *lpq* program checks to see if a daemon process exists for *printer* and prints the file *status*. If the daemon is hung, a super user can use *lpc* to abort the current daemon and start a new one.

waiting for *host* to come up

This indicates there is a daemon trying to connect to the remote machine named *host* in order to send the files in the local queue. If the remote machine is up, *lpd* on the remote machine is probably dead or hung and should be restarted as mentioned for *lpr*.

sending to *host*

The files should be in the process of being transferred to the remote *host*. If not, the local daemon should be aborted and started with *lpc*.

Warning: *printer* is down

The printer has been marked as being unavailable with *lpc*.

Warning: no daemon present

The *lpd* process overseeing the spooling queue, as indicated in the "lock" file in that directory, does not exist. This normally occurs only when the daemon has unexpectedly died. The error log file for the printer should be checked for a diagnostic from the deceased process. To restart an *lpd*, use

```
% lpc restart printer
```

7.3. LPRM

lprm: *printer*: cannot restart printer daemon

This case is the same as when *lpr* prints that the daemon cannot be started.

7.4. LPD

The *lpd* program can write many different messages to the error log file (the file specified in the **lf** entry in *printcap*). Most of these messages are about files which can not be opened and usually indicate the *printcap* file or the protection modes of the files are not correct. Files may also be inaccessible if people manually manipulate the line printer system (i.e. they bypass the *lpr* program).

In addition to messages generated by *lpd*, any of the filters that *lpd* spawns may also log messages to this file.

7.5. LPC

could't start printer

This case is the same as when *lpr* reports that the daemon cannot be started.

cannot examine spool directory

Error messages beginning with "cannot ..." are usually due to incorrect ownership and/or protection mode of the lock file, spooling directory or the *lpc* program.

GED PLOTTING

1. INTRODUCTION

There are two separate ways that hardcopy can be done from GED; VGB mode, which uses the Video Graphics board to rasterize drawings for output to a Versatec, and HPR mode, which uses a separate program running under UNIX to format drawings for output to various plotters. Both of these modes are explained in detail in this document.

In addition to the two hardcopy modes, there is one other facet to GED hardcopy: PLOTTALK. PLOTTALK is a Plotter Interface Access Library. It is a set of C library routines and C data structure set up to allow users to interface to plotters that are not supported by Valid. PLOTTALK is explained in detail in a separate PLOTTALK document.

The last section of this document contains information to help you if you have trouble getting hardcopy to work.

2. HISTORY

Initially, all plotting was accomplished with the Video Graphics (VG) board hardcopy. Because there were three CPU boards, (VG board, Peripheral Interface Board (PIB), and 68000 board) and three sets of code (VG driver code, PIB driver code, and GED) involved, it was very easy to break, and very hard to fix. The VG board would talk directly to the PIB (where the Versatec is attacted) with no direct intervention by the UNIX operating system.

Because of the difficulties with VGB mode hardcopy and with the continued demand for new features, HPR mode was introduced. HPR mode uses a separate program that runs under UNIX on the 68000; it doesn't have as many complicated interactions with system level code, so it's a ideal place to add new features like more plotters, new fonts, color support, etc. Because it's a separate program, HPR also allows standalone hardcopy, and easier porting to new machines.

The 8.0 release of 4.2BSD UNIX provides network spooling support within a homogeneous 4.2 network.

3. VGB MODE HARDCOPY

VGB mode is implemented in GED and VGB code. VGB in local mode (i.e., **not** "set spool") works in the following manner: GED sends scaling information and the display list to the VG board. The VG board rasterizes the list into bits and then run-length encodes the bits to save space and time. Then GED gets the encoded bits and sends them down a pipe to the program *Vpr*. *Vpr* is a shell script wrapper that calls *lpr* with a **-Pvplot** option. **lpr -Pvplot** contacts the *lpd* daemon and gives it the plot data and the printer name ("vplot"). *lpd* makes another copy of itself and tells the new copy to put the encoded bits into the directory listed in */etc/printcap* for the *vplot* printer (*/u0/spool/Vpd*). If the lock file for the directory is locked, (which indicates a daemon is already running) the new copy of *lpd* dies, if not, it locks the lock file and starts sending each of the files in the spool directory to the printer. If the printer is local, *lpd* sends the files to the */dev/vplot* device (which is the entry point to the PIB driver that knows how to handle run-length encoded drawings). If the printer is remote, it sends them to the *lpd* daemon on the remote machine. After all of the files in the spool directory have been send and removed, the daemon unlocks the lock file, and dies.

If GED is in "set spool" mode, then instead of sending the run length encoded bits to *Vpr*, it sends them to the file *vw.spool* (set spool mode is used when you want to save hardcopies online).

When the VG board is busy rasterizing the display list, GED's screen is taken over, and you can't do any useful work. For this and other reasons, VGB mode will only be worked on if bugs are found. All new hardcopy features are added to HPR mode. (See below.) VGB mode only supports Versatec plotters attached to S32s.

4. HPR MODE HARDCOPY

HPR mode is implemented with GED, *hpr*, *lpr*, *lpd*, and *hpf* code. HPR in local mode (i.e., **not** "set spool") work as follows: GED sends *hpf* header information (plotter type, scale, etc.; see section on "hpf file format") and the binary file to be plotted to *hpr*. *hpr* is a shell script wrapper that calls *lpr* with a **-Pgedplot** option. As in the above VGB mode, the *lpd* daemon is contacted and the data is sent to the spool directory listed in */etc/printcap*, (*/u0/spool/hpd*). The */etc/printcap* entry for **gedplot** also says that *lpd* should run the filter program */usr/lib/hpf* on the data. *hpf* transforms the input data into a format that the plotter can understand. It then calls *lpr* with the **-P*plotter_name*** option to send the formated plotter file to the correct directory for the type of plotter that the user specified. Another daemon then spools the plotter-specific files to the plotter.

This double spooling in HPR mode is done so that other programs (like GED using VGB mode) can accsess plotters in an orderly fashion; so that slow pen plotters don't hold up faster Versatecs; and so GED users don't have to wait for there plots to be processed.

If GED is in "set spool" mode, then instead of using *hpr*, GED uses *hpf* with the **-f** option to create the *vw.spool* file. Note that the *lpr* spooling system is not involved, so you must wait while *hpf* creates the *vw.spool* file.

5. HOW TO SET UP THE PRINTCAP FILE

The following sections outlines how to set up the */etc/printcap* files on various machines for network plotting and printing. The emphasis is on the specifics that you need to know to get VALID's printers working. For a more detailed and complete description of the spooling system see the *4.2BSD Line Printer Spooler Manual* included with the 8.0 Change Package and the *printcap*(5) man page.

VALID provides an example of a printcap file in */etc/printcap.proto*. This file contains information on how to set up entries for most printers supported by VALID. Copy this file to */etc/printcap* and modify it for your site.

5.1. THINGS TO REMEMBER

- There must be one */etc/printcap* entry and one spool directory for each different type of printer or plotter you want to use.
- The first entry with a name that matches the one being looked for will be used.
- If your trying to print remotely, there must be a corresponding local printer entry on the remote machine, and the remote machine's */etc/hosts.equiv* file must include an entry for your machine.

5.2. ENTRY FOR A LOCAL SERIAL LINE PRINTER

A basic entry in the */etc/printcap* file is for a local line printer. An example follows:

```
# serial printer over a tty line.
lp|zippylp|Any RS-232 serial printer:\
        :lp=/dev/lp:\
        :br#9600:\
        :tr=\f:\
        :if=/usr/lib/lpf:\
        :sd=/u0/spool/lpd:\
        :st=/u0/spool/lpd/status:\
        :lf=/u0/spool/lpd/errorlog:
```

The *lp* "capabilty" is the device name to open for output. *br* is the baud rate. *tr* is the trailer string to print when the queue is emptied. (In this case, a form feed so that the output can be removed from the printer.) *if* is the output filter. *sd* is the spool directory. *st* is the status file. *lf* is the error log file.

This entry also serves as the server end of a remote printing system. That is, other machines could use this local printer because of this entry.

5.3. ENTRY FOR A REMOTE LINE PRINTER

The entry for a remote printer is as follows:

```
# remote line printer.
lp|remotelp|Prints on zippy's line printer:\
        :lp=:\
        :rm=zippy:\
        :rp=lp:\
        :sd=/u0/spool/lpd:\
        :st=/u0/spool/lpd/status:\
        :lf=/u0/spool/lpd/errorlog:
```

rm is the remote machine name. *rp* is the remote machine's name for the printer. Note that the *lp* entry is empty. It **must** be empty for remote printing to work. You should have either a local or remote entry for line printing, not both.

5.4. ENTRY FOR GED PLOT RASTERIZATION SPOOLER

This entry is used by the HPR mode in GED. A gedplot entry **must** be present for HPR mode hardcopy to work.

```
gedplot|Used by HPR mode:\
        :lp=/dev/null:\
        :if=/usr/lib/hpf:\
        :sd=/u0/spool/hpd:\
        :st=/u0/spool/hpd/status:\
        :lf=/u0/spool/hpd/errorlog:
```

if is the output filter. *lp* is /dev/null because the filter program *hpf* spools the plotter input file instead of sending it directly to the plotter.

5.5. ENTRIES FOR VERSATEC DEVICES

There should be at least two entries for the Versatec in */etc/printcap,* one entry for the Versatec text device(*/dev/vprint*) and at least one entry for the graphics device (*/dev/vplot*). Versatecs will not work without these entries. (This is in addition to the above gedplot entry.)

5.6. ENTRY FOR THE VERSATEC TEXT DEVICE

The local Versatec text device (*/dev/vprint*) entry looks like this:

```
vprint|local 11" Versatec printer:\
        :lp=/dev/vprint:\
        :sd=/u0/spool/vpd:\
        :st=/u0/spool/vpd/status:\
        :lf=/u0/spool/vpd/errorlog:
```

An entry for the remote Versatec printer would look like this:

```
vprint|remote 11" Versatec printer:\
        :lp=:\
        :rm=zippy:\
        :rp=vprint:\
        :sd=/u0/spool/vpd:\
        :st=/u0/spool/vpd/status:\
        :lf=/u0/spool/vpd/errorlog:
```

5.7. LOCAL VERSATEC GRAPHICS DEVICE

A local graphics device (*/dev/vplot*) entry should look like this:

```
vplot|vers11|local 11" Versatec plotter:\
        :lp=/dev/vplot:\
        :sd=/u0/spool/Vpd11:\
        :st=/u0/spool/Vpd11/status:\
        :lf=/u0/spool/Vpd11/errorlog:
```

The entry should have two names in the first line. The first is the generic name ***vplot***. The second should be one of the names that GED and ***hpf*** have for various printers. Known vplot printers are:

vers11	(11" Versatec)
vers22	(22" Versatec)
vers36	(36" Versatec)
vers42	(42" Versatec)
B9424	(24" Benson)

If you ask for an 11" Versatec in GED, the rasterization program (hpf) looks for a "vers11" entry. If it can't find one, the plot will not come out.

NOTE: There **must not** be any spaces between the vertical bars that seperate printer names and the printer names themselves.

5.8. REMOTE VERSATEC GRAPHICS DEVICE

A remote graphics entry follows the pattern outlined above:

```
vplot|vers11|remote 11" Versatec plotter:\
        :lp=:\
        :rm=zippy:\
        :rp=vplot:\
        :sd=/u0/spool/Vpd11:\
        :st=/u0/spool/Vpd11/status:\
        :lf=/u0/spool/Vpd11/errorlog:
```

You can have more than one remote Versatec graphics entry if there is more than one size of Versatec on your network. Remember that you must have a seperate spooling directory for each type of printer or plotter. Another Versatec entry might look like this:

```
vplot|vers36|remote 36" Versatec plotter:\
        :lp=:\
        :rm=godzilla:\
        :rp=vplot:\
        :sd=/u0/spool/Vpd36:\
        :st=/u0/spool/Vpd36/status:\
        :lf=/u0/spool/Vpd36/errorlog:
```

5.9. PEN PLOTTERS

An entry for an HP7580 pen plotter:

```
hp7580|local D size HP pen plotter:\
        :lp=/dev/hp:\
        :sd=/u0/spool/hp7580:\
        :st=/u0/spool/hp7580/status:\
        :lf=/u0/spool/hp7580/errorlog:
```

An entry for an CalComp 1043 pen plotter:

```
    calcomp1043|local E size CalComp pen plotter:\
        :lp=/dev/calcomp:\
        :sd=/u0/spool/calcomp1043:\
        :st=/u0/spool/calcomp1043/status:\
        :lf=/u0/spool/calcomp1043/errorlog:
```

Remote entries can be created as in the above Versatec examples.

6. HPF FILE FORMAT

hpf can read ASCII vectorized format files, ASCII component (body) files, or GED binary format files. The file that *hpf* reads has a special header prepended to it that has information about the type of plotter, line width, scale, and the format of the graphical information. In addition, if the file is binary, there is a list of **rooted** paths of SCALD directories to search to find bodies. The format of the *hpf* input file is:

- **The type of plotter**, followed by a newline character. Currently the plotter type can be: "vers11" "vers22" "vers36" "vers42" "vers72" "calcomp1043" "calcomp5744", "hp7580", "hp7475", or "B9424" The "vers" are various widths of Versatecs, calcomp1043 is a pen plotter, calcomp5744 is an electrostatic plotter, hp7580 is a D size pen plotter, hp7475 is a B size pen plotter, and the B9424 is a Benson 9424 plotter.

- **The line weight**, and *optionally the font type*, followed by a newline. Line weight can be '1' for NORMAL_WEIGHT lines or '2' for HEAVY_WEIGHT lines. The default is '2.' The font type can be "vector_font," "valid_font," "milspec_font," gothic_font," "cursive_font," "greek_font," or "symbol_font." The default is "vector_font." Examples of the fonts can be found in appendix A.

- **The scale**, and *optionally the coordinates per inch* of the drawing coordinates system followed by a newline. The scale can be either an ASCII positive real number string or a capital letter drawing page size ('A', 'B', 'C', 'D', or 'E'). Illegal scales default to '1.0.' The coordinates per inch parameter is an ASCII integer. If the coordinates per inch parameter is not included, any real number scale is assumed to be pre-compensated for the target plotter's bits per inch coordinate system (i.e., hpf multiplies the incoming coordinates by the scale to get correct plotter coordinates).

- **The type of encoding used**, followed by a newline. The type can be 'V,' 'B,' or 'C.' 'V' is vectorized, 'B' is binary, 'C' is component (body). An illegal type causes a fatal error.

- **Only** if the encoding type is binary ('B'), a newline terminated list of **rooted** SCALD-directory paths follows the above encoding type.

- **The encoded drawing.**

6.1. EXAMPLES OF HPF HEADERS

```
vers11                              (11" Versatec)
2 vector_font                       (HEAVY lines, vector font)
1.0 500                             (scale 1 to 1, 500 GED coordinates/inch)
B                                   (binary format file)
/u0/pdh/pdh.wrk                     (scald directories to search for bodies)
/u0/lib/standard/standard.lib
/u0/lib/lsttl/lsttl.lib

----

hp7580                              (HP 7580 pen plotter)
1 milspec_font                      (NORMAL weight lines, milspec font)
D                                   (scaled to D size)
V                                   (Vectorized format file)

----

calcomp1043                         (CalComp 1043 pen plotter)
1                                   (NORMAL weight lines, default font)
E                                   (scaled to E size)
B                                   (binary format file)
/u0/pdh/pdh.wrk                     (scald directories to search for bodies)
/u0/lib/standard/standard.lib
/u0/lib/lsttl/lsttl.lib
```

Note that comments **can not** appear in an hpf header. They are included above to make this document easier to read.

6.2. HOW TO RUN HPR STANDALONE

The above information on hpf's input file format allows you to do hardcopy without starting up GED. A step-by-step illustration of how to do standalone hardcopy follows:

1. Create an hpf header file with all the information on plotter, font, scale, file format, etc. In this example it is called *bsizevers11.header* and contains the following:

   ```
   vers11
   2 gothic_font
   B
   B
   /u0/pdh/pdh.wrk
   /u0/lib/standard/standard.lib
   /u0/lib/lsttl/lsttl.lib
   ```

   Note that there **must not** be any extra lines after the last SCALD directory path or hpf will get confused!

2. You now need to get the name of the binary drawing file that you want to plot. In most cases it's easy to guess the name of the directory that contains your drawing. "MY LOGIC DRAWING" would be in the directory "mylogicdrawing." If you have many similarly named drawings look in your "user.wrk" file to get the mapping between GED drawing names and UNIX directories. In each directory there will be a "logic_bn.*version#.page#*" which is the binary format GED file. In this example the binary file is **mylogicdrawing/logic_bn.1.1**

3. Now you are set to make plots, just type:

```
% cat bsizevers11.header mylogicdrawing/logic_bn.1.1 | hpr

OR

% cat bsizevers11.header mylogicdrawing/logic_bn.1.1 > hpr.infile
% hpr  hpr.infile
```

7. TROUBLESHOOTING HARDCOPY PROBLEMS

The first thing to do if you have trouble with hardcopy is try to decide whether the problem is hardware or software. There are several things you can do to gather information to help you decide.

7.1. HPR MODE ERROR LOG FILE

If you are having problems with HPR mode hardcopy you may be able to find out what the problem is by looking the the HPR mode error log. The error log is in the file */u0/spool/hpd/hpferrorlog*. This file contains the last 3K bytes of error messages from the *hpf* program. Each invocation of *hpf* starts with:

New invocation of /usr/lib/hpf: *date*

After the header, error messages (if any) will appear. Common problems are: can't find a body; permission problems; etc. Also, if there are bugs in *hpf* or GED, there will be internal error messages that will help pinpoint the problem. If you include these internal error messages in a bug report, it will make it much easier for Valid to fix the problem.

7.2. LPD ERROR LOGS

The master *lpd* daemon has an error log */u0/spool/masterlog* for error messages. If you're tring to fix remote printing, look at *masterlog* on the remote machine as well. Each copy of *lpd* for individual spool directories opens an error file if there is a *lf* entry in */etc/printcap*, (usually *spool_dir*/errorlog.) Check these for any error messages.

Here are a few other things to check:

- The owner of all the spool directories should be *daemon.*
- The group of all the spool directories should be *daemon.*
- The mode of all the spool directories should be drwxrwxr-x (i.e., 775)
- The *rp* (remote printer) capabilty on the **local machine** should have an exact match with a **name** of a printer in the */etc/printcap* file of the **remote machine.**
- There must **not** be any spaces between the vertical bars that seperate printer names and the printer names themselves in */etc/printcap.*
- If you are tring to get HPR mode hardcopy in GED to work; remember, the name of the printer in */etc/printcap* must exactly match one of the names HPR mode knows about:

"vers11"	11" Versatec
"vers22"	22" Versatec
"vers36"	36" Versatec
"vers42"	42" Versatec
"Cvers42"	42" Color Versatec
"B9424"	24" Benson
"hp7580"	D size HP pen plotter
"hp7475"	B size HP pen plotter
"calcomp1043"	E size CalComp pen plotter
"calcomp5744"	E size CalComp electrostatic plotter

- Every machine that uses *lpr* must have a "**printer**" service entry in the file */etc/services,* **even for local printing**. Also, all of the */etc/services* files must agree on the sevice number for "**printer**" in order for remote printing to work.

  If you change any of the above, you should kill off the lpd daemon and restart it:

  ```
  # ps ax
  (get process id of lpd)
  # kill -15 process_id
  # /usr/lib/lpd
  ```

You can run the program "ps" to get information about *p*rocess *s*tatus of processes running currently on your machine:

ps -ax

You'll get output something like this:

```
PID     TT    STAT TIME   COMMAND
0       ?     D    8:25   swapper
1       ?     I    1:35   /etc/init
2       ?     D    0:00   pagedaemon
3       ?     D    0:11   rpc_server_ac
33      ?     S    18:05  /etc/update
36      ?     I    22:10  /etc/cron
60      ?     I    0:00   /etc/rshd
63      ?     I    0:00   /etc/rlogind
80      ?     I    0:02   /etc/wm 0 1
84      ma    I    0:35   -csh (csh)
1867    ma    T    0:07   make hpf
5699    00    I    0:22   -csh (csh)
17465   p0    S    0:11   /etc/rlogind
17466   p0    S    0:15   -csh (csh)
17516   p0    R    0:13   ps ax
```

Make sure there is at least one *lpd* running. If not type:

```
# /usr/lib/lpd
```

While you are root.

8.0 SOFTWARE ANOMALIES

9 January 1986

8.0 SOFTWARE ANOMALIES

The following anomalies have been identified in the 8.0 release of the SCALDsystem software. The anomalies are grouped according to program and, where possible, include "workarounds."

SCREEN/HARDCOPY DISPLAY OF DATA

Problem: Characters may be dropped during editing with the VAX/VT100 window using the pass-through port.

Workaround: Set the baud rate to no higher than 2400 baud.

Problem: The **vprint** command collapses tabs in the ascii file to be printed.

GRAPHICS EDITOR

Problem: Copying a group and concatenating the copy to the original can modify the original group.

Workaround: Bear in mind that **COPY** works on vertices, not objects.

Problem: Deleting or moving bodies leaves holes in the grid, especially when **GRID DOTS** is used. Also, a hole can appear in a body that has been moved off the grid. The problem is visual only, not functional.

Workaround: Refresh the screen with the **WINDOW;** command.

Problem: Placing a body outside the borders of the grid may crash GED.

Workaround: Turn on the grid to see the extent of the GED screen.

Problem: Maximum zoom in the lower left corner may cause GED to crash.

Problem: Using the **WIRE** command to create a 4-way tie may later cause the **MOVE** command to treat the wires as unconnected segments.

Workaround: Put a dot at the center of a 4-way tie.

Problem: The update package for the 8.0 Reference manual incorrectly lists the pathname for /u0/scald/section/section as /usr/bin/section. The correct pathname is /u0/scald/section/section.

Problem: A beta version of 7.25 GED produced corrupted versions of ASCII body files.

Workaround: Any drawings produced with that version can be read into a later release of GED and written back out again.

Problem: In the **CHANGE** command a ^s moves the cursor backwards if the cursor was after the character it is searching for.

Workaround: Use ^f to move forward.

Problem: If a group is **CUT** and **DELETEd** and then a **SHOW PINS** is done the pins from the deleted group are still shown.

Problem: If **SET CAPSLOCK_ON** and go into vi (during the **CHANGE** command) the signal names typed in lower case are not changed to caps.

Workaround: Be sure to type changes in upper case.

Problem: Can't swap default properties.

Problem: The following causes the signal name attached to a wire to disappear:

1. **SPLIT** a wire and body (signal attached to wire)
2. **MOVE** the wire
3. **UNDO**

Problem: Hardcopy can only find a drawing in the current use directory.

Workaround: Use the command **HARDCOPY** <scald_dir> drawing_name[;]

Problem: A **CUT** and **PASTE** causes signal names attached to pins to disappear. Also throws away default properties.

LIBRARIES

Problem: BACKANNOTATION causes the pin numbers on a drawing to be different from the notes on the corresponding body. The numbers do not affect connectivity file formats. Section swaps cause a wider divergence in numbering.

Workaround: The problem can be disregarded without impairing functions. If the numbering is confusing, it is possible to create a body without subscripts on the part.

COMPILER

Problem: Due to Pascal implementation differences, a compilation that is successful on a Valid S32 may not compile on an IBM 370 VM machine. The difficulty occurs when the compilation surpasses the IBM virtual memory capacity while processing drawings. When the memory is exceeded, neither the CMPEXP.DAT or CMPSYN.DAT files are created on the 370 VM machine.

Workaround: Use **sepcomp** and **seplink** whenever virtual memory is exceeded and one encounters a limit on the maximum drawing size that can be used on an IBM 370 VM machine.

Problem: Low asserted signals for the Compiler, Timing Verifier, and the Simulator are indicated by a leading "-" instead of a trailing "*" in the listing files. The Compiler, Timing Verifier, and the Simulator currently report the polarity of a signal, not its assertion. The Compiler interprets the assertion and passes both -A and A* on to other tools as -A; similarly, A and -A* are passed on as A.

The following anomalies exist with the new mechanism ValidSIM using the ROOT_DRAWING directive.

Problem: The **PRIMITIVE** compiler directive is not implemented.

Problem: The **SUPPRESS** compiler directive works only on compiler messages. Linker oversights and warnings cannot be suppressed. This applies to the WARNINGS OFF and OVERSIGHTS OFF directives as well.

Problem: The linker will not report signals synonymed to their own complement.

Problem: Error messages not directly associated with a page (such as "Drawing not found") are reported only to the monitor (and cmplog.dat), and are not included in the list file (cmplst.dat) produced by COMPERR.

Problem: The compiler does not detect different property attributes (such as PERMIT(PIN) INHERIT(PIN)) when deciding whether to recompile a page. (Property attributes describe how a property can be used, not what its value is -- see the SCALD manual for a full description.) This kind of change is usually made very infrequently. If changes of this kind are made, then all drawing can be forcibly marked "dirty" by going to all working directories (and library directories) and executing the command **rm** */schema in each. The absence of this file from a drawing directory causes all pages of the drawing to be recompiled.

Problem: The mechanism used to detect changes to plumbing drawings is not entirely foolproof. In particular, if a different "standard" library is specified than was used the last time the page was compiled while leaving the old library files in place this change will not be detected.

TIMING VERIFIER

Problem: The Timing Verifier, at present, cannot interpret user-defined syntax in the case file.

Workaround: Consult explanation above in the Compiler section.

Problem: In the case file, if more than one subscript for a signal is used, only the first bit subscript will be accepted.

Workaround: For a signal, use one bit subscript at a time.

Problem: Using the 40105B part from the CMOS library causes the 7.25 Verifier to crash.

PACKAGER

Problem: The Packager does not guarantee optimal assignment of logical to physical parts.

Problem: The Packager may assign sections of a HAS_FIXED_SIZE part into more than one package when doing feedback.

Problem: When doing feedback on a part-type that has been modified by Physical Part Tables, the Packager requires that the new part_type name, such as RES-1, be given instead of the original part type name RES.

Workaround: Give the modified part_type name when doing Feedback Netlist. The new part_type name for the modified parts can be found in PSTXPRT.

Problem: The name of a body that has been modified by Physical Part Tables is given in the new format. For example, the Packager backannotation file may contain:

```
BODY='LS00-1'
BODY='LS00'
```

Workaround: If doing backannotation after running the Packager with PHYSICAL_PART_TABLE, the user must edit the backannotation file **PSTBACK** to delete the **'-1'**.

Problem: If a section does not have all its pins listed, the Packager ignores the section without doing feedback on it.

Workaround: Make sure each pin of a feedback section is listed when doing Feedback Netlist.

Problem: Packager does not strip blanks preceding LOCATION values. A LOCATION=' U12', for example results in a part named ' U12' instead of 'U12'. This may cause subsequent programs to crash.

Workaround: Watch out for unintentional blanks preceding a LOCATION value.

Problem: Physical part names may contain non alpha-numeric characters (if assigned by the user-assigned LOCATION property), causing DIAL to crash.

Workaround: Don't give a LOCATION value containing non alpha-numeric characters.

Problem: The Packager crashes when there are many parts in the Compiler Expansion File with the same PATH_NAME.

Problem: If a new section is allocated to a package using the LOCATION property, the Packager may reassign the sections already assigned to this part rather than leaving them as they were.

Workaround: Rather than using the LOCATION property to assign a new section, reassign sections manually for sections whose location and section are important.

Problem: If a comment is typed on two lines during entry in the physical part table, a syntax error results.

Workaround: Confine all comments for this table to one line.

Problem: The use of tabs within comment lines for the physical part table results in syntax errors.

Workaround: Do not use tabs in the comments for the physical part table.

Problem: If a net contains only bidirectional nodes, the net-input-loading is reported as zero.

Problem: When the file feedback_netlist is used for pin swapping, the Packager does not report an error for the swapping of two pins with different PIN_GROUP property values.

Workaround: Check to see that pins to be swapped by feedback have matching PIN_GROUP property values in the library. After packaging, check the output files to be certain that the pinswap was done properly.

SIMULATOR

Problem: Non-existent signals in the Tabular input file may later cause the Simulator to crash.

Workaround: The Simulator outputs a message when non-existent signals are discovered. Remove the non-existent signals and their values from the file before using it. This problem will be fixed in ValidSIM 2.0.

Problem: Tabular input files cannot be read in the split screen GED/Simulator.

Workaround: Use the stand-alone Simulator when using the tabular input features.

Problem: There are various problems associated with the **PATCH** command, particularly when the user tries to re-PATCH a PATCHed signal.

Workaround: The cycle time to make a GED change and restart the Simulator has been dramatically reduced in ValidSIM 1.0. Keep the use of PATCHing to a minimum.

Problem: It is possible for the Simulator to go into an infinite loop while evaluating a circuit containing PASS TRANSISTOR or RES (resistor) primitives. In addition, it is not possible to deposit to a bidirectional net.

Workaround: Insert a buffer into the circuit at the input(s) to the RES or PASS TRANSISTOR primitive. Values can then be deposited to the input of the buffer rather than directly to the RES or TRANSISTOR.

Problem: If a PASS TRANSISTOR is connected to a bubbled output, the simulated behavior may not be correct.

Workaround: If fully bi-directional gates are not required in the circuit, use the UNI PASS TRANSISTOR primitive rather than the PASS TRANSISTOR.

Problem: The user should not attempt the **trace_all** procedure on large designs.

Problem: Simulating for excessive intervals (> 30,000,000 ns) with one SIM command can cause the Simulator to crash.

Workaround: Simulate in more reasonable intervals - break huge intervals into smaller ones (e.g., instead of "SIM 50000000" use "SIM 25000000" twice). This problem will be fixed in ValidSIM 2.0.

Problem: If the user types ^C to abort the new graphics Simulator, ^Z is disabled when the user returns to UNIX and certain other functions may not work correctly (e.g., the editor **vi** operates incorrectly).

Workaround: The ^Z function can be restored by entering the UNIX command **stty susp ^Z**. This problem will be fixed in ValidSIM 2.0.

Problem: If the user types ^Z or ^C to stop the simulator, the screen will not scroll correctly if the terminal type was set to GCLUSTER and is now set to some other terminal type.

Workaround: The scrolling region can be reset by entering the UNIX command **echo '<ESC>[r'**. This problem will be fixed in ValidSIM 2.0.

Problem: When using Realchip with Realfast, if the user specifies "clock_edge=rise" or "clock_edge=fall" in the Realchip definition file, the simulated behavior may not be correct.

Workaround: Change the specification to "clock_edge=both".

Problem: In Realfast, if a constant signal is connected to a pin of the RESISTOR primitive, the constant value is not propagated to the other pin of the RESISTOR (e.g., a pull up resistor will not pull the pin up to 1).

Workaround: Insert a buffer between the constant signal and the resistor, where the input is the constant signal and the output drives the resistor.

Problem: The clock period is not appropriately scaled by the RESOLUTION factor if the RESOLUTION directive appears after the CLOCK_PERIOD directive in the Simulator directives file.

Workaround: When using the RESOLUTION directive, ensure that it appears before the CLOCK_PERIOD directive in the directives file. This problem will be fixed in ValidSIM 2.0.

Problem:Contrary to the description in the manual, simple breakpoints cannot be **CLEAR**ed by name. This problem will be fixed in ValidSIM 2.0.

Workaround: Use the number of the breakpoint to **CLEAR** it.

Problem: Typing ahead ruins the display, particularly when the terminal type is set to GCLUSTER.

Workaround: Do not type ahead of the Simulator. Wait until the "*" prompt is displayed before entering commands.

Problem: The Simulator does not generate a screen image properly in the list file when the **SNAPSHOT** command is executed under GED or when running as GCLUSTER.

Workaround: Use the **HARDCOPY** command to generate a screen image. **HARDCOPY** will generate a graphical image which is not possible in the ASCII list file.

Problem: Inconsistent or out of range parameters in the **MEMLOAD** command can cause the Simulator to crash.

Workaround: Make sure that memory locations exist before you try to load them; make sure bit ranges and/or word ranges match when specifying **MEMLOAD** parameters. This problem will be fixed in ValidSIM 2.0.

Problem: The **DUMPMEMORY** command causes an error message to be output when running on the IBM.

Problem: The **ASSERTIONS** command does not accept timing data in the split screen GED/Simulator.

Workaround: Use the stand-alone Simulator when using the **ASSERTIONS** command. This problem will be solved in ValidSIM 2.0.

INTERFACES

Problem: The Scicards interface cannot handle a Scicards LATCH4 function unless precautions are taken. Otherwise, the result is an error ''209): Scicards pin not found on the SCALD part.'' This condition causes the program to end.

Workaround: Do not attach the SCI_PART property to library parts which are modelled by LATCH4 functions in the SCICARDS library. Omitting this property ensures that an entry will be generated in the scilib.dat file for that part.

COMMUNICATIONS

Problem: When rlogining onto a non-VALID system the first login request you get will not be successful as a result of some date that gets sent to the window.

Workaround: In response to the first login request hit a carriage return. Enter login name to second request.

Problem: When rlogining onto a non-VALID system where you have an account, some data gets sent to the window. As a result the first couple of commands entered will be unrecognized.

Workaround: Hit a couple of carriage returns before trying to type any commands.

Problem: When rlogining in to another system, rlogin sometimes repeats bursts of characters on the screen. This does not cause any loss of data.

Problem: Ftp will core dump if it attempts to write to a full disk on another VALID system.

UNIX KERNEL

Problem: **iostat** does not report disk information correctly because the new Rimfire disk driver in 8.0 does not update **/dev/kmem** correctly. However, other statistics reported by **iostat** are correct.

Problem: Catching **SIGSEGV** is fatal to the system.

Workaround: Do not catch **SIGSEGV**.

COMBINED RESIDENT MONITER

Problem: The system hangs when using the *coredump* command in **crm** on a *DEC* "vt100" terminal or *Decwriter* hardcopy terminal.

Workaround: The problem is that **CRM** doesn't know how to generate a carriage return for a *vt100* terminal. When the data from the coredump command reaches the end of line, hit the carraige return manually.

LANGUAGE COMPILERS

Problem: A Fortran program that has been compiled on a system with software before 7.0 produces the message ''illegal instructions'' and dumps the core.

Workaround: Recompile the program with 7.0 or later software.

Problem: The C Compiler gives the message:

"Switch table overflow"

if more than 248 cases exist in a single switch statement.

Problem: The C Compiler refuses to cast a function to type ''void,'' thereby creating an infinite loop.

Workaround: Either delete the keyword *void* or use a typedef to equate it to int (or some other legal typename).

Problem: Limit on size for single arrays in C.

Workaround: Do not create any single arrays larger than 32K.

Problem: The comma operator in a C program argument list is erroneously evaluated.

Workaround: Avoid using this construct.

Problem: Compiling anything with !n where n is double (not long) causes the Compiler to do a core dump.

Workaround: Do not use !n. Use (0==n) instead. Cast the variable to long before using the not operator.

Problem: If units are defined in a procedure in the interface section but are not defined in a body in the implementation section, a Pascal program can be compiled without complaint, but its executed version crashes.

Workaround: Define everything in the implementation section.

Problem : **Ranlib** not supported.

Workaround: Use lorder and tsort to order library modules before archiving. For example, the following command lines will order and archive object modules whose names are listed in "file1" into the library "newlib.a":

```
lorder `cat file1` |tsort > file2
ar cv newlib.a `cat file2`
```

Problem: Fopen modes "w+", "r+" and "a+" are not supported.

Problem: Locv routine in libc.a doesn't work.

Workaround: Use iocv(3).

DR-11

Problem: Filecopy can't transfer Scald directory filenames with special characters in them. For example, filenames containing "_" are not recognized on the Vax.

Workaround: Rename directories before transferring them.

Problem: When copying files from a Valid system to the Vax the owner of the file is not given delete permission.

Problem: Filecopy hangs if Vax is down.

Workaround: Check status of Vax before transferring files.

RSCS/VM

Problem: On occasion RSCS crashes and the restart program does not always work.

Problem: RSCS messages follow the log-in name through multiple windows and appear on the one that the user has most recently logged-in on, not necessarily the current window. Some messages, consequently, may not be read.

Workaround: Create a special user for the purpose of sending RSCS data. Logon as that user to use RSCS transfer. Then the messages will also go to the window using the RSCS login.

Problem: The RSCS commands are not documented on the on-line manual.

Workaround: See the Valid RSCS User's Manual.

Problem: GETRDR changes all files in a CMS disk to variable length format. The problem occurs when a file with file name of * and file type of * is sent to the VM machine, which does not define * as a legal file name or file type. The

EXEC uses this combination while generating a rename instruction; consequently, all files are changed.

Problem: The soft key assignments for PF Keys 1 through 9 are inoperable within the 3277 emulation mode. PF Keys 10 through 12 do work, and in XEDIT mode all keys function. The problem is in the simulation code on the S-32.

Problem: RSCS installation doesn't include rscs%mail command.

Problem: IBM file names received by Valid host are truncated to 14 characters.

Problem: RSCS has problems establishing handshake at 57.6 KB.

Problem: If a user has more than one active window RSCS sends messages to the last window. This also may cause RSCS to hang.

Problem: RSCS will not work in single user mode.

UNIX UTILITIES

Problem: The table formatter *tbl* sometimes dies on legal tables due to a doprnt bug.

Problem: The **tar cv /u0** command does not work if run from /u0 or a subdirectory of /u0.

Workaround: Change to the root position and then type **tar cv u0** to archive the data. Within the /u0 directory, the command **tar u0/*** also works.

Problem: The **ps -au** command sometimes erroneously reports users with more the 100% usage of CPU resources. The problem typically occurs during heavy system load.

Problem: The **adb** utility does not assign the correct breakpoints for two byte instructions. For example, any C program that has been compiled into a.out can then be checked by **adb**. Once within **adb**, the user can give the following instructions:

```
_start+ Ox16:b
:r
:c
:c
```

and find that segmentation is at _start+ Ox1c. Consequently, the breakpoint is set at most two byte instructions.

Problem: Using **adb** in conjunction with wraplib.o is difficult. If one compiles C and Fortran or Pascal together, and then wishes to set a breakpoint at "open" (as called by C) in the resulting a.out, the command

```
open:b
```

sets a breakpoint at _open, not "open."

Problem: Using **cat** to display a binary file on a design station can cause unpredictable results that require the user to reload the Video Graphics Board.

Workaround: It is helpful to use the **file** command to identify binary files before attempting to **cat** any file. However, **file** does not correctly identify the file type for core files, and a user should not **cat** a core file without using the **strings** procedure first and then using **cat** to examine the output of the **strings** program. If you are using the C shell, try aliasing the **cat** command to **cat -v**.

Note that the **ls -F** command indicates directories with a slash mark and executable files with an asterisk. Many users find it helpful to **alias ls** to **ls -F** in the .cshrc file.

Problem: During **walknet** procedures, the **tp** program reads only MTA0 instead of the logical device designated by the command file.

Workaround: Use **tp -f** to force **tp** to force the program to use the first-named file rather than tape as the archive.

Problem: If the user specifies both the user name and the terminal number while using the **write** command, the message is sent to that terminal regardless of whether or not another user is logged in.

Workaround: Use the **who** command to find out who is logged in before sending a **write** message.

Problem: A line printer cannot be set for 2400 baud with the **stty** command. The code for the line printer daemon is hard-coded for 9600 baud.

Workaround: Put the following command into the rc.local file:

```
(stty 2400; sleep 10000000) < /dev/lp > /dev/lp 2>&1 &
```

Problem: The C shell terminates whenever the user types in 'echo' followed by the name of a non existent command in backquotes (‘ ‘) The machine responds with a segmentation error message and a core dump and logs the user out.

Workaround: Make sure the command exists before using it in conjuction with echo.

Problem: The syslog command in **sendmail(8)** is not supported.

Problem: Killing the **lpd** daemon with "-9" (hard kill) leaves the socket "/dev/printer" lying around on the root filesystem.

Workaround: Use "kill -15" (soft kill) to terminate **lpd**. This allows **lpd** to unlink the socket before dying. NOTE: **/etc/rc** removes the socket while bringing the system up multi-user. Make sure you use the utility **shutdown** to bring the system down or the socket will be left around.

Problem: Tar cv /etc doesn't pick up all the files in the /etc directory.

Workaround: Use tar cv /etc/* instead.

Problem: The csh returns a "bus error" if a command line is more than 256 characters long.

Workaround: Don't make command lines over 256 characters long.

Problem: The csh dies if history substitutions create a command line longer than 256 characters.

Workaround: Try to avoid substitutions which will result in very long command lines.

Problem: **more** and **page** don't work on files containing less than 3 characters.

Workaround: Use **cat**.

Problem: The **dd** command fails if the "files" option is greater than 1.

Problem: su doesn't export the user's variables in the bourne shell.

Workaround: After su do a **. .profile**. This will export the variables set up in the .profile file.

Problem: **Vi** doesn't work correctly on a vt100 terminal if the **autowrap** option is set.

Workaround: Turn the autowrap option off.

SCALDstar

Problem: If an object is very large relative to the scale factor, the object may disappear. That object can include the cursor.

Workaround: Zoom out enough to make the object reappear.

Problem: **remove** command in LED does not work correctly across the net.

Workaround: Use a local LED to remove drawings.

Problem: Dynamically dragging an even number of objects, all of which share an identical edge, across the screen causes the coinciding line to disappear. Occasionally, small bits of the ghost image are left on the screen. This description pertains only to 8.0 **led** release.

Workaround: Redraw the screen to eliminate any bits of images. The problem is purely visual during the act of moving the objects and in no way affects the functionality of the objects once they are deposited in a new location.

Problem: DRC may report some spacing errors in addition to the width errors during a width check if minimum spacing is less the minimum width for a layer.

Workaround: Ignore those errors. All real errors are correctly reported.

Problem: Cells with too many (>10,000) cannot be converted to LED format.

Workaround: Subdivide cell between conversion.

NOTES FOR 8.0 SOFTWARE RELEASE

9 January 1986

NOTES FOR 8.0 SOFTWARE RELEASE

The following contains notes relevant to the 8.0 release of the SCALDsystem software. Notes are grouped by program.

SCREEN/HARDCOPY DISPLAY OF DATA

1. The vgb mode is the default hardcopy mode in GED for monochrome workstations.

GRAPHICS EDITOR

1. Using **SET CAPSLOCK_ON** passes upper-case letters to the Simulator and to the UNIX operating system. If you do use **SET CAPSLOCK_ON**, all UNIX commands need to be issued from a separate window and all script files for the Simulator need to have upper-case letters.

2. GED maps SCALD directory file names to lower case. In the event that you use mixed case letters for directory file names, an error message appears. It says that GED cannot open a file name with the name in lower-case letters and also says that there is no SCALD directory by that name (in upper-case letters). As a workaround, do not create file with upper-case letters in the pathname.

3. The current algorithm for naming unnamed nets can cause a problem if a part is deleted from an unnamed net. If a part that was used to generate the unnamed signal name is deleted, then GED changes the name of the net. The renaming is local, and only the altered net is renamed, not the entire database.

   Note that these net names are part of the database for physical design systems such as Scicards. For further information, see the relevant section of the Software Changes Information document.

4. If a monochrome display user draws a large circle, uses the **window;** command for a section on the upper right hand side of the screen, and then moves the circle around, the pixels in the GED screen, including the menu, are turned off. The difficulty can wipe out other windows. This procedure should not be attempted.

5. The vgb mode is the default hardcopy mode in GED for monochromatic systems.

6. There is no way to turn off the display of the number of bodies, properties, etc. that appear in the upper left corner of the screen while executing a group command.

7. In rare instances, the asterisk (*) lags behind by one string while using control-X in the **CHANGE** command. Refresh the window by issuing the **WINDOW;** command.

8. By default, the Graphics Editor provides a 0.61 text size for backannotated pin numbers in order to make a distinction between numbers that pertain to pins and other numbers/text on the drawing. To change the size to 0.8, use the **find $pn** command, followed by **DISPLAY 0.8** group_name.

9. A body with a pin at the origin cannot be rotated or changed into different versions. Do not place a pin at the coordinates (0,0).

11. The user cannot abort the **DIRECTORY** command when the output is more than one screenful. See instructions in the manual for the use of metacharacters in combination with the **DIRECTORY** command.

COMPILER

1. The compiler does not detect different values for the BUBBLE_CHECK, SUPPRESS, WARNINGS or OVERSIGHTS directives when deciding whether to recompile a page.

COMMUNICATIONS

1. GED cannot be started with a remote login, since GED has to run on a local Video Graphics Board.

2. Network communications commands (**rlogin**, **rsh**,, etc.) require that both systems have compatible kernels. For instance, version 6 software systems can only communicate with version 6; version 7.0 through version 7.25 can communicate with each other; and version 7.27 through 8.0 can communicate with each other.

UNIX KERNEL

1. The UNIX system clock is s software clock which should not be depended upon for accurate time. It's purpose is to provide a time stamp for files.

2. Using the **Berknet** line discipline can cause the system to crash a program tries to do an *ioctl(2)* call after the **Berknet** line discipline has been selected. Use the old tty line discipline, "OTTYDISC", or the new tty line discipline, "NTTYDISC", instead of the Berknet discipline, "NETLDISC".

RSCS/VM

1. Be sure the bisync and rdaemon processes have been killed before restarting.

2. RSCS does not start if there is no entry in /u0/spool/rscs with the system machine name as a directory. If the entry is omitted, the system generates the warning message "rdaemon: cannot initialize -- could not initialize MSGWORK" in errorlog. To workaround, make a directory in /u0/spool/rscs with the system machine name. A VM system expects upper-case letters for machine names.

3. Transfer of binary files to IBM host is not supported by RSCS.

UNIX UTILITIES

1. The **ls -c** command operates as **ls**.

2. The disk usage command (**du**) reports only the number of blocks in the size of directories, not the number of kilobytes as stated by the documentation. The actual disk storage used may be considerably less than **du** reports.

3. The tutorial program learn is not on the system due to insufficient space on the disk.

4. The program /etc/mknetusr is no longer supported on the system. The program /etc/mkusr now will duplicate a password entry from a remote system onto a current system. See mkusr(8V) for more details.

ANOMALIES CLOSED AS OF 8.0 FINAL

9 January 1986

ANOMALIES CLOSED AS OF 8.0 Final

The following anomalies have been fixed in the 8.0 release of the SCALDsystem software. They are listed according to program.

SCREEN/HARDCOPY DISPLAY OF DATA

1. Deleting a window that still has a process in it, particularly a background process, causes problems with the continuing processes and with subsequently-created windows.

2. The emulation of the visual editor **vi** scrolls one line at a time during moves through a file.

3. VT100 emulation doesn't work correctly. (Refer to the 8.0 Changes Document for more information.)

GRAPHICS EDITOR

1. GED does excessive canonicalization during an edit, thereby slowing the system down.

COMMUNICATIONS

1. Entering ^D to *tftp* prompt causes infinite loop

2. The *tftp* daemon is not supported by *ind.*

3. Typing a control-C during a remote login sometimes generates the message ''Lost connection.''

4. When **rsh** takes a long time to execute and the user issues a control-C, a forked process is left on the remote machine and the local shell has an unusual stty state.

5. The **rsh** command gives an ''address already in use'' error in certain instances. For example, **simlab** may generate that message after an **rsh simlab date** command.

UNIX KERNEL

1. The large (Bourne-shell) scripts for software installation at customer sites randomly fail.

LANGUAGE COMPILERS

1. The **ulinker** program causes a core dump if it is given more than 128 .obj files. The file names, in this case, are entered interactively by giving no command line arguments to /u0/fortran/ulinker.

2. Lockpages library routine in libc.a doesn't work

UNIX UTILITIES

1. The Graphics and Layout Editors ignore the umask setting in the C-Shell.

2. Online manual page for getdtablesize should be named getdtablesize.2.

3. Mkusr doesn't handle Cntrl-C very well.

4. Using the **touch** command without arguments causes a core dump.

5. All user .login, .profile and .cshrc files are owned by root and have the permission code set at -rw-rw-rw.

6. The procedure for using **restor xv** directory_name has problems with restoring selected files or directories from a multi-volume dump tape. It tries to restore files onto parent directories that do not exist and then prints the message "unable to create file." The **restor** program has been replaced in 8.0 by **restore(1)** which handles this problem.

7. Rebooting in single user mode can cause a file that was being edited to come up empty after a **reboot -s** is issued.

8. The **dump** procedure may generate an error message about a bad unit number or block number. For example, typing

   ```
   dump 0u /dev/rim0b
   ```

   produces the error message:

   ```
   Rim 01: Bad unit   or block   , block 71759
   ```

   and a core dump.

9. Using control-Z in the Bourne Shell while in the single-user mode hangs the system.

10. The table formatter **tbl** for **nroff** is not on the system.

11. The **uucp** utility is not supported. See the 8.0 document, UUCP OPERATION AND MAINTANENCE.

12. The system accounting utilities (**sa**, **accton**) are not supported.

13. The on-line manual for Section 8V UNIX commands can be called to the screen only if an upper-case V is included as part of the command name. Typing **man 8V ether** is successful, but **man ether** and **man 8v ether** results in the message ''not found.''

14. Using the **ms** macro package with nroff or troff causes the following lines to be displayed along with the text output:

```
.asciz "$Header: tmac.s,v 800.0 85/07/31 17:55:36 root Exp $"
.asciz "%W% %Y% %Q% %G%"
```

15. If a system manager specifies a non-existent file system as a user's home directory during the **mkusr** program, the program terminates and creates an erroneous entry in /etc/passwd. For example, a single-disk system has /u0 as the only file system; **mkusr** terminates if a user then specifies /u1.