AT&T

UNIX® System V/386

UTILITIES RELEASE NOTES

UNIX® System V Release 3.0
INTEL 80386 Computer Version

Utilities Release Notes

PRENTICE HALL, *Englewood Cliffs, New Jersey 07632*

IMPORTANT NOTICE TO USERS

While every effort has been made to ensure the accuracy of all information in this document, AT&T assumes no liability to any party for any loss or damage caused by errors or omissions or statements of any kind in the UNIX® System V/386 Utilities Release Notes © AT&T, its upgrades, supplements, or special editions, whether such errors are omissions or statements resulting from negligence, accident or any other cause. AT&T further assumes no liability arising out of the application or use of any product or system described herein; nor any liability for incidental or consequential damages arising from the use of this document. AT&T disclaims all warranties regarding the information contained herein, whether expressed, implied or statutory, including implied warranties or merchantability or fitness for a particular purpose.

AT&T reserves the right to make changes without further notice to any products herein to improve reliability, function or design.

Printed in the United States of America

10 9 8 7 6 5 4 3 2

ISBN 0-13-936121-9 025

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

NOTICE

The information in this document is subject to change without notice.
AT&T assumes no responsibility for any errors that may appear in this document.

386/ix is a trademark of Interactive Systems Corporation.
ACT is a trademark of Micro-Term.
AnnArbor is a trademark of AnnArbor Terminals.
Beehive is a trademark of Beehive International.
Concept is a trademark of Human Designed Systems.
CrystalWriter is a trademark of Syntactics.
DATASPEED is a registered trademark of AT&T.
dBASE II is a registered trademark of Ashton-Tate.
DEC, PDP, VAX, and VT100 are trademarks of Digital Equipment Corporation.
DOCUMENTER'S WORKBENCH is a trademark of AT&T.
Dataphone is a registered trademark of AT&T.
Develcon is a trademark of Develcon Electronics, Incorporated.
Diablo is a registered trademark of Xerox.
Dow Jones News/Retrieval Service is a trademark of Dow Jones.
Ethernet is a registered trademark of Xerox.
HP is a registered trademark of Hewlett-Packard, Inc.
IBM is a trademark of International Business Machines.
IMAGEN is a trademark of IMAGEN Corporation.
INFORMIX is a registered trademark of Relational Database Systems.
INGRES/CS is a trademark of Relational Technology.
INSTRUCTIONAL WORKBENCH is a trademark of AT&T.
Intel is a registered trademark of Intel Corporation.
LSI is a trademark of Lear Siegler.
MBASIC is a registered trademark of Microsoft.
MICOM is a registered trademark of MICOM System, Incorporated.
MS-DOS is a registered trademark of Microsoft Corporation.
MULTIBUS is a registered trademark of Intel Corporation.
Micro-Term and MIME are trademarks of Micro-Term.
Microsoft is a registered trademark of Microsoft.
Multiplan is a registered trademark of Microsoft.
Official Airline Guide is a trademark of Official Airline Guide, Inc.
PC-Interface is a registered trademark of Locus Computing.
Penril is a trademark of Penril Corporation.
RM/COBOL is a trademark of Ryan-McFarland.
SuperCalc3 is a trademark of Sorcim/IUS Micro Software.
Syntactics is a trademark of Syntactics.
TEKTRONIX and TEKTRONIX 4010 are registered trademarks of Tektronix, Inc.
TELETYPE is a registered trademark of AT&T.
TeleVideo is a registered trademark of TeleVideo Systems.
Teleray is a trademark of Research Inc.
TermiNet is a trademark of General Electric.
UNIX is a registered trademark of AT&T.
UltraCalc is a trademark of OLYMPUS Software.
Unify is a registered trademark of Unify.
Ventel is a trademark of Ven-Tel, Incorporated.
Versatec is a registered trademark of Versatec Corporation.
WE is a registered trademark of AT&T.
WRITER'S WORKBENCH is a trademark of AT&T.
Weitek is a trademark of Weitek Corporation.
XED is a trademark of Computer Concepts.
Xenix is a registered trademark of Microsoft Corporation.

AT&T Products and Services

- To order documents from the Customer Information Center:
  - within the continental United States, call 1-800-432-6600
  - outside the continental United States, call 1-317-352-8557
  - send mail orders to:

    AT&T Customer Information Center
    Customer Service Representative
    P.O. Box 19901
    Indianapolis, Indiana 46219
- To sign up for UNIX system or AT&T computer courses:
  - within the continental United States, call 1-800-221-1647
  - outside the continental United States, call 1-609-639-4593
  - TELEX: 1-609-639-4756
    Attention: Training Registration
- For information on Intel hardware and software, contact the Intel sales office nearest you.
- To find out about UNIX system source licenses:
  - within the continental United States, except North Carolina, call 1-800-828-UNIX
  - in North Carolina and outside the continental United States, call 1-919-855-2737
  - or write to:

    Software Licensing
    Guilford Center
    Salem Bldg. 4th Floor
    P.O. Box 25000
    Greensboro, NC 27420

Table of Contents

List of Figures

Table of Contents

Table of Contents

Preface

These *Release Notes* contain important information about UNIX System V Release 3.0 on your computers. First, they briefly describe the new features of this release. Software notes, additional information about installation, and compatibility notes are also provided. finally, these *Release Notes* list the documents that pertain to Release 3.0.

For a comprehensive description of the software and documentation available for UNIX System V Release 3.0 for your computers, see the *Product Overview*. For a complete description of the procedures used in the administration of your computer running UNIX System V Release 3.0, see the *System Administrator's Guide*.

Additional Products

Two products of UNIX System V Release 3.0 are offered separately: Remote File Sharing and the Networking Support Utilities. For a description of software installation and build procedures, software notes, and information about documentation for these products, see the *Remote File Sharing Release Notes* and the *Networking Support Utilities Release Notes*.

Conventions Used in these Release Notes

In this document, as in all UNIX System documentation, certain typesetting conventions are followed when command names, command line format, files, and directory names are described. There are also conventions for displays of terminal input and output.

- You must type words that are in **bold** font as they appear.
- *Italicized* words are variables; you substitute the appropriate values. These values may be file names or they may be data values, as applicable.
- CRT or terminal output and examples of source-code are presented in `constant-width` font.
- Characters or words in square brackets, [ ], are optional. (Do not type the brackets.)

A command name followed by a number, for example, **ed**(1), refers to that command's manual page, where the number refers to the section of the manual. Manual pages from section (1) appear in the *UNIX System V User's Reference Manual,* unless otherwise noted. Manual pages from sections (3) and (4) appear in the *Programmer's Reference Manual.* Manual pages from section (1M) appear in the *System Administrator's Reference Manual.*

Examples in these *Release Notes* show the default system prompt for UNIX System V, the dollar sign (**$**). They also show the default prompt when you log in as super-user, the pound sign (**#**).

These *Release Notes* refer to packages and to products. A package is a group of programs that do related things. For example, the Editing Utilities package contains UNIX System V's text editors and their associated files. UNIX System V Release 3.0 comprises many packages. A product is something that you purchase independently of UNIX System V Release 3.0, for example, Remote File Sharing.

Features of UNIX System V Release 3.0

This release maintains the compatibility of source code across the 80386 computer family (with the exceptions noted below), and object code between 80286 and 80386 computers at the application level. Application packages, such as DOCUMENTER'S WORKBENCH Software, INSTRUCTIONAL WORKBENCH Software, and WRITER'S WORKBENCH Software, continue to work with Release 3.0.

UNIX System V Release 3.0 provides the following new features. For a more detailed description of them, see the *Product Overview*.

- Remote File Sharing (optional product)
- Networking Support Utilities (optional product)
- Enhanced Basic Networking Utilities
- Shared Libraries
- Command Syntax Standard
- Signal Mechanism Enhancements
- Improved Terminal Support Facilities
  - Terminal Information Utilities Enhancements
  - AT&T Windowing Utilities Package

  In addition, the following new features are added for the Intel Architecture.
  - 80286 Binary Compatibility
  - Floating Point Coprocessor Support

- Additional Features
  - **help** Facility Extensions
  - **crash**(1M) Command Changes
  - New System Header Files

Remote File Sharing

To take advantage of Remote File Sharing, you must have Remote File Sharing Utilities, the Networking Support Utilities, and a transport provider. (See "Networking Support Utilities" below for a description of a transport provider.) These optional products require at least 2 megabytes of main memory.

Remote File Sharing lets you share files, directories, devices, and named pipes transparently among computers that are linked by a network. Files are shared transparently by mounting a remote directory as one would mount a file system. Each computer on the network controls which local resources are available to other computers and which remote resources local users may access. For example, with Remote File Sharing you may share a directory among several departments of your business, or a letter-quality printer or typesetter that no one department could support by itself. For more information, see the *Remote File Sharing Release Notes* and the *System Administrator's Guide*.

Networking Support Utilities

The Networking Support Utilities is an optional product that requires at least 2 megabytes of main memory.

The Networking Support Utilities provide STREAMS tools, the AT&T Transport Interface, and the Listener. STREAMS is a set of tools for development of communication and networking services within the UNIX system; the Transport Interface is based on the Transport Service Definition (Level 4) of the International Standards Organization (ISO) Open Systems Interconnection

(OSI) reference model, and defines how a user accesses the services of a transport protocol. The Listener receives requests for network services from another system, interprets which network service is needed, and starts a process that has been named to provide the requested network service. The Listener then drops out of the communications path and continues to listen for new service requests.

For more information about the Networking Support Utilities, see the *Networking Support Utilities Release Notes*; for more information about the Listener, read these release notes and the *System Administrator's Guide*.

Enhanced Basic Networking Commands

Media-independent basic networking commands are provided in the Basic Networking Utilities package, but they require the optional Networking Support Utilities product to be used.

Basic networking commands [for example, **uucp**(1C) and **uux**(1C)] have been enhanced to conform to the AT&T Transport Interface (see the *Networking Support Utilities Release Notes* for details). These utilities can communicate using any Transport Provider that conforms to the AT&T Transport Interface. The operation of the Basic Networking Utilities commands is the same regardless of whether you install the Networking Support Utilities product.

The Networking Support Utilities can show which, if any, Transport Providers are available with **nlsadmin -x**. If you install additional Transport Providers, no changes are needed to the software to accommodate the underlying media or protocols; you need only register Basic Networking Utilities services with the network Listener for those Transport Providers, and register the Transport Providers in the administrative files for the Basic Networking Utilities [see **nlsadmin**(1M) in the *System Administrator's Reference Manual*].

Even with these enhancements, the syntax of the basic networking commands is no different from before. See the *System Administrator's Guide* for details about how to manage this facility.

Shared Libraries

The Advanced Programming Utilities product and the C Programming Language Utilities product allow the programmer to build a shared library of routines accessed at run-time, rather than having those routines combined with an application program at load time.

On Release 3.0 systems, two shared libraries are available: the most commonly-used routines from the C Library and the Networking Services Library. Most of the UNIX system commands use these two libraries, as do any applications that were built with them. For more information about generating Shared Libraries, see the chapter devoted to this topic in the *Programmer's Guide.*

Command Syntax Standard

getopt(1) allows shell procedures to parse command lines to check for legal options and to process option arguments. A new command, **getopts**(1), is an enhanced version of the **getopt**(1) command. **getopts** is consistent with and supports rules 3 through 10 of the UNIX system command syntax standard. [The standard is described on the **intro**(1) manual page.]

You may use **getopt** in shell scripts with UNIX System V Release 3. However, you should use **getopts** instead of **getopt**; beginning with the next major UNIX System V release, **getopt** will no longer be supported.

To assist in the conversion of shell scripts that are affected by a change from **getopt** to **getopts**, a conversion command, **/usr/lib/getoptcvt**, is provided. (See the **getopts** manual page for details.)

Signal Mechanism Enhancements

A new set of system calls [see the **sigset**(2) manual page in the *Programmer's Reference Manual*] provides a mechanism to catch and hold signals without losing them during later processing and to guarantee that a process reaches the signal handler before it is interrupted by another signal.

Some additional signal-handling features, provided by other popular operating systems, are also available.

Improved Facilities for Supporting Terminals

Support for terminals is improved with new features of the Terminal Information Utilities and the new AT&T Windowing Utilities.

Terminal Information Utilities

The "Terminal Information Utilities" package (often called **curses/terminfo**) has the following new features:

- expanded support for terminal filters, soft labels, and new AT&T terminals
- new commands: **captoinfo**(1M) converts **termcap** entries to **terminfo** entries; **infocmp**(1M) compares two **terminfo** entries or prints entries in several formats. [Section (1M) is in the *System Administrator's Reference Manual*.]
- new options to the **tput**(1) command to initialize, reset, and learn the "long name" of a terminal
- an improved version of the **terminfo** compiler, **tic**
- new documentation on the manual pages and in the "**curses and terminfo**" chapter of the *Programmer's Guide* (Chapter 10)

These new **curses** features are only available with the 3.0 version of **curses** on UNIX System Release 3.0. [See also **curses**(3X) in the *Programmer's Reference Manual*.] All programs that ran with System V Release 2 **curses** will run with System V Release 3.0. You may link applications with object files based on the Release 2 **curses/terminfo** with the Release 3.0 **libcurses.a** library. You may link applications with object files based on the Release 3.0 **curses/terminfo** with the Release 2 **libcurses.a** library, as long as the application does not use the new features in the Release 3.0 **curses/terminfo**.

AT&T Windowing Utilities

This new package contains software that is required by AT&T windowing terminals, such as AT&T's line of DMD terminals (for example, the 5620). Routines included are for creating, deleting, and manipulating terminal windows, querying terminal window status, and providing statistics about usage

of windowing routines. For more information about this package, see the **layers**(1) manual page, the **libwindows**(3X) manual page in the *Programmer's Reference Manual*, and Appendix F, "Setting Up Your Terminal," in the *User's Guide*.

Additional Features (help, crash, encryption)

UNIX System V Release 3.0 provides additional information in the **help** facility, improvements to the **crash**(1M) command, and repackaging of encryption mechanisms.

help Facility Extensions

Descriptions and examples of many additional commands, terms, and symbols have been added. See **help**(1), **glossary**(1), **starter**(1), and **usage**(1).

crash Command Changes

In addition to providing debugging support for the new operating system features included in Release 3.0, **crash** has been changed extensively to make it easier to use. The syntax of all the functions has been standardized so that similar functions share similar syntax. There is a **help** function within **crash**, a number base converter, a memory search function, and a disassembler capability. The **crash** (1M) manual page in the *System Administrator's Reference Manual* describes the details of this new **crash** command.

New System Header Files

New header files were added to **/usr/include**: **unistd.h** [definitions for symbolic constants introduced and used throughout the /usr/group Standards document; see **unistd**(4) in the *Programmer's Reference Manual*] and **limits.h** [definitions for commonly used values that vary from implementation to implementation, see **limits**(4) in the *Programmer's Reference Manual*]. Several new definitions were added to the header file **/usr/include/sys/stat.h** to make it easier for programmers to write portable code.

286 Binary Compatibility

The 286 binary compatibility allows the 286 binary executables files compiled under UNIX System V/286 Releases 2.0 and 3.0 using the SGS's provided in those systems. These 286 binary files can be loaded onto the system and executed just like 386 binary files. 286 binary compatibility is subject to the following restrictions:

- The 386 *sdb*(1) debugger cannot debug a 286 program because of format differences between 286 and 386 executable files and processes.
- The *ptrace*() system call is not supported. It is used only by debuggers, and is not necessary for running applications.
- *prof*(1) does not report execution profiles.
- System V/286 Release 3.0 shared libraries are not supported.
- The effective maximum number of open files when a 286 program is exec'ed is one less than NOFILES. NOFILES should be configured to at least 21.
- The number of segment descriptors available in the LDT for 286 programs is limited to 256 unless LDTSZ is increased. This is sufficient to map 16 MB of text and data. LDTSZ is defined in **sys/seg.h**, and is not a configurable parameter, because the end of the LDT could run over the end of the U block if LDTSZ were set too high.
- A 286 program may require more stack space running on the 386 than when running on a 286 because of increased system call and signal-handling overhead.
- The following *ioctl*(2) commands defined in **sys/termio.h** are supported. They are TCGETA, TCSETA, TCSETAW, TCSETAF, TCSBRK, TCXONC, and TCFLSH. Other commands can be supported by adding code in **i286emul/Sioctl.c** and rebuilding the emulator.
- The 286 executable file must be readable by the user that execs the program (not simply executable).

Floating Point Coprocessor Support

This release provides support for Intel 287/387 and Weitek WTL-1167 floating point coprocessors. It also provides 387 and WTL-1167 emulation when the corresponding coprocessor is absent. The system can be configured into one of the following six coprocessor configurations:

	387 Emulation	287/387
No WTL-1167 Support	1	2
WTL-1167	3	4
WTL-1167 Emulation	5	6

The kernel at boot time checks the presence of the chip and automatically configures the systems accordingly except when WTL-1167 is absent. In that case, the system administrator needs to determine if Weitek emulation is needed and make a new kernel accordingly. The kernel can be built from binary by deleting or adding the module 'weitek' in the system file under directory $CONF/systems. For details, refer to the section on "Instructions for Making Kernels". Source code changes can be made in /usr/src/uts/i386/ml/misc.s for detection of the 387 and 1167 for OEM systems that do not use the Intel floating point detection mechanism.

The C compiler generates code for the 287/387 coprocessor, but not for the WTL-1167. Independent software vendors have compilers for WTL-1167 code generation. Please contact your local Intel sales office. However, the WTL-1167 math library is included in the release and can be called from the C compiler.

Installation and Boot Procedure

The UNIX System V/386 Release 3.0 source code is available in both 1/4-inch cartridge tape and 9-track tape. The C compiler is also available as a separate package in both 1/4-inch cartridge tape and 9-track tape. The source code supports both Multibus 1 (MB1) and AT-based(AT386) 386 systems. The content of the source code is described in the "Source Code Provision Release Notes".

Two versions of binaries, one for an Intel MB1 system and one for an AT386 system, are available. Please contact your local Intel sales office for information about how to obtain copies. This section describes the installation and boot procedures for both versions of binaries.

Remote File Sharing and Network Support Utilities are not currently packaged as separate products. They are installed as part of the UNIX System V Release 3.0. For first time setup and startup instructions, see the Remote File Sharing Utilities Release Notes and the Networking Support Utilities Release Notes.

Installation

The installation of the system is self–explanatory. Read through these instructions completely, read the enclosed manual pages, and see the attached errata sheet for corrections before attempting to install the final release. This installation procedure will destroy all previous contents of your master disk. If you are already running a Beta or earlier release, see the section on "Updating Beta Systems" in this document.

Multibus 1 (MB1)

The binary release for an Intel Multibus I system consists of one binary installation tape (1/4-inch cartridge). An Intel Multibus I system must have at least the following equipment to boot this release:

1. A 386/2X single board computer.
2. At least 2 MB of memory.

3. The 386 chip revision level of at least B1.
4. The i214 disk-tape controller.
5. DEBUGMonitor PROMs (verison D011 or later).
6. An asynchronous ASCII terminal set to 9600 baud.
7. At least 40 Mb of hard disk storage compatible with the controller.
8. A cartridge tape drive.

You should power on the machine and all associated peripherals and follow one of the two console sequences depending on the level of boot EPROMs in the system:

First console sequence:

```
DEBUGMonitor - 386 d011
[....]
Copyright 1986 Intel Corporation.
>
```

Second console sequence:

```
After the console displays "xxxx", enter <Shift>U.
After system diagnostics and the following display:

break to DMON-386 Monitor (y or [n]) ?

Answer 'y' to this in 30 seconds.
(otherwise it times out to iSDM Monitor;
then you need to restart from the beginning)

DEBUG Monitor - 386 d016 (or d015)
[....]
Copyright 1986 Intel Corporation
>
```

Insert the release tape and type:

b ":wta0:"

Note that the double-quote marks and the leading and trailing colons are required. Typing errors may be corrected by using the backspace key or the <DEL> key. Note also that leading and trailing blanks are significant to some

of the interactive programs that ask you for a response. When you are asked to type (y/n) for yes or no, type exactly y<RETURN> or n<RETURN>.

A special version of the UNIX system on the tape will be booted. It uses a RAM-disk to contain the small set of programs required to install UNIX. The RAM-disk contains (among others) the following programs:

/bin/sh
/bin/cpio
/etc/mkfs
/etc/mkpart
/bin/mv
/bin/mkdir
/etc/disksetup
/INSTALL

After "mounting" the RAM-disk, the second-stage installation will proceed automatically. The remainder of the final release software is contained in a *cpio* archive on the same cartridge.

AT386

The AT386 binary release consists of fifteen 386 binary diskettes. An AT386 system must have at least the following equipment to boot this release:

1. The 386-chip revision level should be at least B1.
2. At least 2.5 MB of 32-bit RAM (preferably 4 MB).
3. IBM AT-compatible disk controller.
4. At least 40 Mb of hard disk storage compatible with the controller.
5. A high density floppy drive.
6. A monochrome, EGA, or CGA display adapter and monitor.

To install on an AT386 system, you will need a boot floppy that has a fully configured UNIX system kernel and the necessary utilities to format your hard disk and install the binaries on it properly. If you don't have an AT boot floppy, you may either contact an Intel Sales Office or you may build your own boot floppy on a Multibus system. The Multibus system must be configured with a 1.2 MB floppy disk drive to build an AT bootable floppy since the 360K diskettes do not have the necessary capacity needed to produce an AT

boot floppy. The rest of this section assumes that you have received a boot floppy that contains a UNIX system kernel linked wth the 386/ix AT Drivers Package.

A set of 15 installation diskettes have been provided to allow you to install the software on most Intel 386 AT compatible systems. The hard disk driver supports any disk that is known in the PC AT ROM BIOS hard disk table. The IBM DOS diagnostics diskette should be used to set the CMOS RAM to the proper value for your disk. You should also use the DOS diagnostics disk to set your memory size to the proper value and you should use the DOS advanced diagnostics disk to format at least the first track of your hard disk if it has never had software installed on it before. Insert the Boot Diskette into the first diskette drive and turn on your AT386 system.

The boot program will load and give you the following prompt:

boot:

If you don't enter anything, after 1 minute **/unix** will boot automatically; if you want to boot **/unix** immediately, enter a carriage return. The install script will be automatically invoked after UNIX has booted. During the boot process you may see a message:

NOTICE: pdinfo does not match parameters.

This indicates that the system is using the disk partition information other than what is provided in the CMOS memory. You can ignore this message.

When the first stage of the install is completed, the diskette-based kernel will shut itself down automatically. To re-boot, remove the diskette from the diskette drive, turn the machine off, and then turn the machine back on. The second stage install will then run automatically. The INSTALL script will ask you to insert the System Installation Disks in order. These diskettes contain the software to be installed on the hard disk.

Software Installation

You will be prompted to confirm that you want to install the system. Type <**return**> to confirm this. The <DEL> key will abort installation. The disk setup program will assist you in configuring your disk. Although it is fairly self-explanatory, the following are recommended:

1. You will be asked to confirm each of your groups of selections so that it is possible to "try again" at most points in the program.
2. To prevent unexpected failures, it is highly recommended that you specify bad blocks so that they can be remapped to alternate sectors on the disk.
3. Ignore the warnings from *mkpart* about not having alternates or boot partitions. The program is run several times and the boot partition is installed in a later run.
4. Installation of additional disks is done after initial configuration. These disks are completely unaffected by the installation.
5. Systems that are heavily loaded can require more than the default allocation of swap space. To change this, answer the question which asks you to confirm the partitioning with n (no), answer the question about separate root and usr file systems with n, (see the first item above), and respond to the question on how many cylinders to use for swap/paging with a larger number (we have used 79).
6. The **root** and **/usr** file systems require at least 34,000 blocks (17.5 Mb) for all of the binaries.
7. If you specify separate **root** and **/usr** file systems, specify at least 100 cylinders for the root (25 more cylinders to include space if **/tmp** is not a separate file system on another disk) and at least 150 cylinders for **/usr**.
8. Do not interrupt the loading of the files unless you plan to start over. You might find your root without any devices at all.
9. Loading files takes about 30-45 minutes.

Boot Procedures

MB1 Boot Procedure

The following list of instructions will boot a System V/386 final release system after the initial installation procedure has been completed.

The first step, after power has been applied to your system, is to get access to the boot PROMs. This may be accomplished by turning the front panel key to reset, and then back to the unlock position. On a modified 386/310 system, this may be accomplished by pressing the front panel "R" (reset) button.

After the system has been reset correctly, the boot PROMs will print a header message on the console notifying the user that the debugger is up, and that it is waiting for commands. The term "DMON" should appear in the header, and the command line prompt should be a ">".

The command to be entered is **b** followed by a carriage return. At this point the UNIX system should proceed to boot up, ask the user for the correct date and time and go into multiuser mode (level 2). The following prompt will then be displayed:

Console Login:

Type **setup** (no password will be required). This special login will prompt you to set the date, time, passwords for various system accounts, and add accounts for new user IDs. It also completes the installation procedure by setting protection modes and ownership of various important devices. The installation procedure requires that you use this log in as the final step. To configure additional disks, log in as root and run **addisk**.

AT386

The following list of instructions will boot a system V/386 final release system after the initial installation procedure has been completed. Apply power to the system and wait a minute or two; the second-stage boot should be loaded automatically by the PROMs and the prompt:

boot:

should appear; if you don't enter anything, after 1 minute **/unix** will boot

automatically. If you want to boot **/unix** immediately, just press the return key. The UNIX System should then proceed to boot up, ask the user for the correct date, and go into multiuser mode (level 2). During the boot process you may see a message:

NOTICE: pdinfo does not match parameters.

This indicates that the system is using the disk partition other than what is in the CMOS memory. You may ignore this message.

Major Device Numbers

AT/386 Only		*Both Systems*		*Multibus I Only*	
hd	block(0)			i214	block(0)
hd	character(0)			i214	character(0)
fd	block(1)			i8251	character(1)
fd	character(1)				
		mem	character(2)		
asy	character(3)			i214tp	block(3)
				i214tp	character(3)
cpyrt	block(4)				
kd	character(5)			ramd	block(5)
				rand	character(5)
		prf	character(6)		
lp	character(7)				
rtc	character(8)				
		log	streamd(9)		
		sp	streamd(10)		
				i552	streamd(11)
		clone	streamd(12)		
		xt	character(13)		
		sxt	character(14)		
				i546	character(15)
		gentty	character(16)		
		osm	character(17)		
cram	character(18)				
urp	streamd(19)				
nau	streamd(20)				
srm	streamd(21)				
		np	block(22)		
		np	character(22)		
		is	streamd(23)		
		pty	character(24)		
		ptx	character(25)		
		it	streamd(26)		
		itd	streamd(27)		

The module names are as follows:

asy*	AT/386 serial ports for tty00 and tty01
clone	streams clones
cpyrt	AT/386 phony driver to write out 386/ix copyright message(s)
cram*	AT/386 battery-back-up CMOS RAM
fd*	AT/386 floppy disk
gentty	indirect tty
hd*	AT/386 Winchester disk
i214	MB1 wini and floppy
i214tp	MB1 streamer tape
i546	MB1 multiport serial/parallel card
i552	MB1 stream driver for Intel 552 Ethernet card
i8251	MB1 single serial port (tty01)
is*	tcp/ip STREAM driver
it*	tcp/ip STREAM driver
itd*	tcp/ip STREAM driver
kd*	AT/386 keyboard and display
ld0	tty line discipline 0
log	streams log
lp*	AT/386 parallel (printer) port(s)
mem	memory (mem, kmem, and null)
nau	AT/386 AT&T Starland network board stream
np*	tcp/ip STREAM driver
osm	UNIX Operating System messages
prf	profiler
ptx*	tcp/ip pseudo tty driver
pty*	tcp/ip pseudo tty driver
ramd	MB1 ramdisk
rtc*	AT/386 real-time clock
sp	stream pipes
srm	AT/386 Starlan network administrative stream
sxt	shell layers
urp	AT/386 Starlan network protocol
xt	BLIT packet multi-tty multiplexer

*These drivers are available through ISV's.
Contact your local Intel sales office.

Naming Conventions for Hard Disk Devices

Hard disk devices are named according to their controller, drive, and partition numbers using the format:

/dev/[r]dsk/c<controller number>d<drive number>s<partition number>

Please see the System Administrator's Guide for descriptions of the uses of partition numbers.

For example, partition 3 on controller 1, drive 1 would be named **/dev/[r]dsk/c1d1s3**, and partition 1 on controller 1, drive 2 would be named **/dev/[r]dsk/c1d2s1**.

On the 386 machines, the first part of the name for disks attached to controller 0 is usually removed. So the first partition (by convention, the entire disk) on controller 0, drive 0 is named **/dev/dsk/0s0** instead of **/dev/dsk/c0d0s0**.

You can check the major number for hard disk devices by checking the number used for other similar devices **(ls -l /dev/dsk/*s*).** On Multibus 386 machines, minor numbers are determined in the following fashion:

minor = board*16 + port
  where
board = position that the board is configured for
      port = port number on board (0..n).

For example (on the Multibus 1 and AT386):

/dev/dsk/0s0 has major# 0, minor# 0
/dev/dsk/0s2 has major# 0, minor# 1
/dev/dsk/0s3 has major# 0, minor# 3
/dev/dsk/0s6 has major# 0, minor# 6
/dev/dsk/1s0 has major# 0, minor# 16
/dev/dsk/1s1 has major# 0, minor# 17
/dev/dsk/1s2 has major# 0, minor# 18

Naming Conventions for Diskette Devices

Diskette devices are named according to the convention:

/dev/[r]dsk/f<drive number><density (d or s or q)><sectors>d[t]

[**t** means the whole disk, no **t** means skip the boot track (Cyl 0)]

so:

f0d9dt double density, 9 sectors per track, the whole floppy
f0d9d double density, 9 sectors per track, all but the boot track
f0q15dt Quad density, 15 sectors per track, the whole floppy
f0q15d Quad density, 15 sectors per track, all but the boot track

The reason for **d** or **dt** at the end of the name is to allow you to have a bootable floppy with a file system after the boot track.

You can determine major numbers for diskette devices by checking the major numbers on existing diskette devices. Only the minor device numbers that were initially installed on your system are currently supported.

To transfer files on a diskette to other machines, invoke cpio as follows:

```
cpio -ocvB >/dev/rdsk/f0d9dt
```

for low density (360 KB) diskettes, or

```
cpio -ocvB >/dev/rdsk/f0q15dt
```

for high density (1.2 MB) diskettes.

The corresponding diskette device names on other systems are:

```
/dev/rdsk/0s24  UNIX System V Release 2, Intel 310/286 system
/dev/dvf0       XENIX Release 3, Intel 310/286 system
/dev/fd0        XENIX Release 2.1 - IBM PC/AT
```

UNIX System V/286 Release 2.0 on the PC 6300 Plus uses the same naming conventions as System V/386 Release 3.0.

Naming Conventions for TTY Devices

TTY devices are named according to the following convention:

ttyi<port number>

Currently, the only exceptions to this rule are **/dev/console** and **/dev/tty01** which refer to ports on the processor board.

If both of the two terminal controller boards are configured on a 386 MB1 system, the ports from the first board are labeled:

ttyi0<port number (0-3 for 546 boards and 0-7 for 188/48 boards)>

and those on the second board are labeled:

ttyi1<port number>

The most important thing to know about tty's is how their major and minor device numbers are determined for use in mknod calls.

The device numbers for MB1 serial I/O ports are determined as follows:

Console port:
/dev/console is a character device with major# 1 and minor# **0**. It is the console port on the processor board.

iSBX351 port:
/dev/tty01 is a character device with major# 1 and minor# 1. It is the iSBX351 port on the processor daughter board.

iSBC546 ports:
minor = board*12 + port
where
board = position that the board is configured for.
port = port number on board (0..n). For the 546, ports 0-3 are the async ports, port 4 is the parallel printer port, and 5 is the calendar "port."

So, for a 546 board that is configured as board 1 (default jumpering), the devices set up are:

```
mknod /dev/ttyi0 c 15 12   ; terminal port 0
mknod /dev/ttyi1 c 15 13   ; terminal port 1
mknod /dev/ttyi2 c 15 14   ; terminal port 2
mknod /dev/ttyi3 c 15 15   ; terminal port 3
mknod /dev/lp    c 15 16   ; parallel port
```

For a 546 board that is configured as board 0, the devices set up are:

```
mknod /dev/ttyi0 c 15 0    ; terminal port 0
mknod /dev/ttyi1 c 15 1    ; terminal port 1
mknod /dev/ttyi2 c 15 2    ; terminal port 2
mknod /dev/ttyi3 c 15 3    ; terminal port 3
mknod /dev/lp    c 15 4    ; parallel port
```

iSBC188/48 ports (configured as board 0):

```
mknod /dev/ttyi0 c 15 0    ; terminal port 0
mknod /dev/ttyi1 c 15 1    ; terminal port 1
mknod /dev/ttyi2 c 15 2    ; terminal port 2
mknod /dev/ttyi3 c 15 3    ; terminal port 3
mknod /dev/ttyi4 c 15 4    ; terminal port 4
mknod /dev/ttyi5 c 15 5    ; terminal port 5
mknod /dev/ttyi6 c 15 6    ; terminal port 6
mknod /dev/ttyi7 c 15 7    ; terminal port 7
```

AT386 port:

```
mknod /dev/tty00 c 3 0     ; COM1
mknod /dev/tty01 c 3 1     ; COM2
```

Naming Conventions for Cartridge Tapes

There are two ways to access cartridge tape devices. The first is /dev/rmt<#>. This will cause the tape driver to rewind when it detects an end-of-file mark (EOF). If you want the tape to remain in its current position after an EOF and NOT rewind, /dev/rnmt<#> should be used.

```
mknod /dev/rmt0   c 3 48    ; rewinds at EOF
mknod /dev/rnmt0  c 3 49    ; does not rewind at EOF
```

Shutdown Procedures

The following steps are necessary to shut down a UNIX V/386 System computer properly:

1. Log in as **root** and change your directory to /.
2. Run the shutdown program with the following options:

   shutdown -y -gTIME

   where TIME is the number of seconds to be allotted before the system is actually halted. A time factor of at least 120 seconds (2 minutes) is recommended for your systems that are being used as multiuser sites. The time factor provides a means to allow users a chance to exit editors and save programs before the system goes down.

The system will proceed to shut itself down after the allotted time, and when the **Reset the CPU to reboot** message appears, the system may be turned off. On an AT/386 System, the usual <CTL> <ALT> <DEL> key combination can be pressed or the system can be turned off and then turned on again in order to reboot at this point. On a Multibus I system, the interrupt button or key can be used, or the power can be turned off and on.

Instructions for Updating Selected Files From the Release

MB1 Release

Occasional disk failures may make it necessary to read in some of the files from the release tape without reading the whole release. To do this, it is necessary to read past the first four files on the release. The first four files of the release are binary images for the bootstrap, the bootstrap operating system, and two RAM-disk images. The *cpio*(1) image of the installed system commands and data follows these.

If you want to load selected files from the release, you need to position the tape to the *cpio* image and *cpio* in whichever files you require. A regular expression can be used to read in whole directories or subtrees. The **-u** option to *cpio* should be used to replace defective executables to assure the file is written regardless of the date in the defective file. This sequence should be performed as **root**. Load the release tape into the tape drive and advance the tape to the end of volume 4 by performing the following sequence of commands:

```
su  root
cd  /
mt  -f  /dev/rnmt0 fsf  4
cpio  -icBvd  </dev/rmt0  regular-expression ...
```

The regular-expression should be the full path name of the files requested, without the initial /. There is no particular order to the files written on the release tape, so you should read the whole tape to make sure you get the files you desire.

For example, to load contents of the directory **/usr/lib/uucp**, you should specify:

```
cpio  -icBvd  </dev/rmt0  'usr/lib/uucp/*'
```

AT386 Release

Diskettes are used instead of cartridge tape. Lists are available for determining which installation diskettes should be searched for the desired files.

Once you determine which diskette contains the file or files that you require, insert the diskette into the drive and type:

```
cpio -icBvd < /dev/dsk/f0q15dt '<filename>'
```

If you require the entire contents of the diskette, type:

```
cpio -icBvd < /dev/dsk/f0qdt
```

Instructions for Duplicating Release Binaries

MB1 Release

The MB1 distribution tape contains the following files:

1. Second-Stage Boot
2. UNIX
3. Install RAMDISK Image
4. Maintenance RAMDISK Image
5. unix

To make a copy of this tape, make sure there is sufficient space on the hard disk, insert the distribution tape and then enter the following commands:

```
ulimit 60000    (#this is 30 MB)
mt -f /dev/rnmt0 rewind
dd if=/dev/rnmt0 of=boot bs=5120
dd if=/dev/rnmt0 of=unix.kernel bs=5120
dd if=/dev/rnmt0 of=install.unix bs=5120
dd if=/dev/rnmt0 of=maint.unix bs=5120
dd if=/dev/rnmt0 of=unix.sys bs=5120
mt -f /dev/rnmt0 rewind
```

Remove the distribution tape and insert a fresh tape. Make sure that the write lock is NOT enabled and then enter the following commands:

```
ulimit 60000    (#this is 30 MB)
mt -f /dev/rnmt0 rewind
dd of=/dev/rnmt0 if=boot bs=5120
dd of=/dev/rnmt0 if=unix.kernel bs=5120
dd of=/dev/rnmt0 if=install.unix bs=5120
dd of=/dev/rnmt0 if=maint.unix bs=5120
dd of=/dev/rnmt0 if=unix.sys bs=5120
mt -f /dev/rnmt0 rewind
```

The distribution tape can also be used as a maintenance tape. This will boot up the machine and load an image of the maintenance part of tape in memory that can be used as a RAMDISK. To boot the tape in maintenance mode, insert the tape in the tape drive, reset the machine and enter:

```
b ":wta0:#fs=1"
```

This will put the machine in maintenance mode with the **root** file system on the RAMDISK.

AT386 Distribution Diskettes

To duplicate a 1.2 MB diskette, use the *dd* command.

Examples:

From diskette:

```
"dd if=/dev/rdsk/f0q15dt of=floppy1 bs=30b count=80"
   or
"dd if=/dev/rdsk/f0q15dt of=floppy1 bs=15360"
```

To diskette:

```
"dd of=/dev/rdsk/f0q15dt if=floppy1 bs=30b count=80"
   or
"dd of=/dev/rdsk/f0q15dt if=floppy1 bs=15360"
```

Use the above commands to duplicate each of the final release binary diskettes.

UUCP Notes

Uucp can be set up to connect to other UNIX system machines from System V/386. Although complete uucp documentation exists elsewhere, the following paragraphs describe the minimum steps required to connect uucp up to another machine immediately.

Assuming the machine name is *spidy*, and the system is connected to tty01 via a directly connected RS232 serial line, the remote login ID is *nuucp*, and the remote password is *panzer*, do the following items:

1. Edit **/usr/lib/uucp/Systems**, and add the line:

   spidy Any spidy 9600 spidy " " \r gin:--gin: nuucp word: panzer

2. Edit **/usr/lib/uucp/Devices**, and add the line:

   spidy tty01 - Any direct

3. Change the mode of **/dev/tty01** to 666 so that it can be accessed by uucp:

   chmod 666 /dev/tty01

Assuming the machine name is *spidy*, and the system is connected to tty01 via a Hayes 1200-baud modem, the phone number is 555-1212, the remote login ID is *nuucp*, and the remote password is *panzer*, do the following items:

1. Edit **/usr/lib/uucp/Systems**, and add the line:

   spidy Any ACU 1200 5551212 " " \r gin:--gin: nuucp word: panzer

2. Edit **/usr/lib/uucp/Devices**, and add the line:

   ACU tty01 - 1200 hayes

3. Change the mode of **/dev/tty01** to 666 so that it can be accessed by uucp:

   chmod 666 /dev/tty01

In System V/386 Release 3.0 it is now possible for uucp to simultaneously share a line with a dialin getty line. This is done through the use of a new program, *uugetty*. After uucp has been set up on a line, for example tty01, the next step is to enable a *uugetty* login on that line. This can be done by editing

/etc/inittab to add the new tty, and then telling *init* to re-read inittab. Do the following:

1. Edit **/etc/inittab**, and add the line:

   ug:23:respawn:/usr/lib/uucp/uugetty -r tty01 1200

2. Tell *init* to re-read the **/etc/inittab** file:

   init q

Things to remember:

1. When connecting the serial lines to your system for a shared dial-in/dial-out line, make sure that the serial lines are capable of dropping DTR. If DTR is not dropped properly, your system may have a tendency to keep switching to a lower baud rate, thus making it harder to log in a second time.
2. *Uugetty* with the **-r** flag does not output anything until it first reads in input. This makes it possible for two systems to share one line and both call each other. It also requires that DTR not be tied high, and that when a user logs into the line, carriage returns must be sent before a login prompt will show up.
3. These are basic instructions only, and the user should read the *System Administrator Guide* for further information.

Instructions for Making Kernels

This section describes how to build new kernels for other MB1 or AT386-based systems.

Note, for AT386 systems: the following procedures make kernels (both from source and binary) and will work if you have obtained the 386/ix UNIX System Device Drivers package. To see if you have these drivers, check for the following files:

source:
cd $ROOT/usr/src/uts/i386/io and look for fd.c, hd.c, cram.c, lp.c, kd.c, rtc.c, asy.c

object:
cd /etc/atconf/modules and look for fd/fd.o, hd/hd.o, ...

For more information on obtaining these drivers, contact your local Intel Sales Office.

Making From Source

To make the kernel from sources, perform the following steps:

1. Set and export the environment variable **$ROOT** to the root of the source tree:

   ```
   ROOT=/usr/myroot; export ROOT
   ```

2. Make the directory **$ROOT/etc** and copy the files **/etc/mkunix** and **/etc/config** there:

   ```
   mkdir $ROOT/etc
   cp /etc/mkunix /etc/config $ROOT/etc
   ```

3. The kernel debugger sources are not included with the UNIX System V.3/386 source tape. To build any kernel, it is necessary to move the kernel debugger directory using the following commands:

   ```
   cd $ROOT/usr/src/uts/i386
   mv db db.d
   ```

After a kernel is built from source, you can build a kernel with the kernel debugger by following the next procedure "Building From Binary". First, copy the **db.o** object to **$ROOT/etc/conf/modules/db** with the following commands:

```
cd  $ROOT/usr/src/uts/i386/db.d
cp  db.o  $ROOT/etc/conf/modules/db
```

The kernel debugger can be configured into the kernel by adding the "db" module to the appropriate system file under **$ROOT/etc/conf/systems**.
See the "Kernel Debuggers" section for more detail.

4. Go to the base source directory:

```
cd  $ROOT/usr/src
```

5. Execute the *:mkuts* script (make sure it is executable). This will build kernel objects and install them in the binary configuration tree, which by default is **$ROOT/etc/conf**, or **$ROOT/etc/atconf** for AT386 systems. (Note: the environment variable $CONF should NOT be set.)

```
:mkuts MB1

   or

:mkuts AT386
```

The default system is MB1. You may also specify a different configuration tree target using the **-c** option:

```
:mkuts AT386 -c $ROOT/etc/conf

   or

:mkuts MB1 -c $ROOT/etc/myconf
```

Also, to configure a kernel that has the kernel debugger in it, append **debugger** to the command line:

```
:mkuts MB1 debugger
```

The default debugger is the kernel debugger. To include the PROM level debugger (DMON), add a "dmon" entry in $ROOT/usr/src/uts/i386/systems/system.std and make the kernel as above.

When specifing the "debug" option on the *:mkuts* command line, the ASSERT statements in the kernel source, used for debugging purposes, will be activated. This is not to be confused with the "debugger" option.

In any of these cases, *:mkuts* will build a kernel configuration tree and run **/etc/mkunix** [see *mkunix*(1M) in the Systems Administrator's Reference Manual] to perform binary kernel configuration. The final kernel will be placed in the configuration tree in the **kernels** directory.

To remove all object files from the kernel source tree, simply run *:mkuts* with the **clean** option:

```
:mkuts clean
```

If you desire to clean out the binary configuration tree as well, use the **clobber** option:

```
:mkuts clobber
```

Thus, for example, to make an MB1 kernel and then make an AT386 kernel with the debugger, simply type the following:

```
ROOT=/usr/myroot; export ROOT
mkdir $ROOT/etc
cp /etc/config /etc/mkunix $ROOT/etc
cd $ROOT/usr/src
:mkuts MB1
:mkuts clean
:mkuts AT386 debugger
```

The end result will be two kernels, one of which resides in **/usr/myroot/etc/conf/kernels**, and the other in **/usr/myroot/etc/atconf/kernels**.

Building From Binary

To reconfigure the kernel from a binary release, perform the following steps:

Set the environment variable **$CONF** and **$MORECPP**.

For an AT386 system,

```
CONF=/etc/atconf; export CONF
MORECPP=-DAT386
```

For an MB1 system,

```
CONF=/etc/conf; export CONF
MORECPP=-DMB1
```

If your configuration tree is rooted in a location other than these defaults, simply set **CONF** to that root. (Note: the environment variable **$ROOT** should NOT be set.)

1. Go to the **systems** directory of the configuration tree:

   ```
   cd $CONF/systems
   ```

   Make a copy of the standard system file:

   ```
   cp system.std system.new
   ```

2. Set the environment variable **SYSTEM** to the name of your new system file:

   ```
   SYSTEM=system.new; export SYSTEM
   ```

3. Edit the new system file to your desires, removing or adding modules or tunable parameters [see *system*(4)]. The names of the modules are listed in directory $CONF/modules. The tunable parameters can be found in either $ROOT/usr/include/sys/kdef.h or $CONF/modules/*/space.c.

4. Run the *mkunix* script:

   ```
   /etc/mkunix
   ```

The new kernel will be built and placed in the directory **$CONF/kernels**. In the preceding example, its name would be **unix.new.1**. In any case, *mkunix* will report the name of the new kernel. More details on the operation of *mkunix* can be found in *mkunix*(1M).

The kernel can also be processed with *mcs*(1) if desired to replace the SCCS identifier strings. The bootable UNIX system kernel distributed with this release may have been processed with *mcs* and may therefore have a different size than a kernel built from the distributed source or binary without any changes.

Configuring System With New Drivers

New drivers can be configured into the system by building new kernels from binary. Before building the kernels, first add the drivers to the kernel configuration tree ($CONF) as follows:

- cd $CONF/modules; mkdir driver_name
- Move all the driver object files into the driver_name directory.
- Write a config file for the driver [see config(1M)] and put it into the driver_name directory.
- If there are tunable parameters, write and install a space.c file for the driver which:
  - #defines the default values of the tunable parameters;
  - #include's config.h, to allow system administrators to override the defaults;
  - allocates space for the variables which depend on the tunable parameters.
- If there are stubs for driver functions which are unconditionally called by the kernel whether or not the driver is present, include these in a stubs.c file in the driver_name directory. This step is only necessary if the kernel has actually been modified by the driver writer.
- Modify the system file in $CONF/systems to list driver_name [see system(1M)].
- Remake the kernel from binary as described in the section, "Instructions for Making Kernels".

Bootloader Software Development

The first-stage bootloader implemented in the PROMS loads in the second stage bootloader and passes control to it. The second-stage bootloader then loads the UNIX system kernel and starts the system initialization. The source code for the second-stage bootloader is located in the source code directory, $ROOT/usr/src/uts/i386/boot. The README files provided in the directory give detailed description of how bootloaders work for both MB1 and AT386 systems.

The second stage bootloader executes in the 8086 real mode of the 386 processor. The 386 C compiler has no support for the 8086 real mode. To develop your own second-stage bootloader, you need to use the 286 C compiler which can be executed on a 386 system under the 286 emulator provided in the release.

To install a 286 C compiler on your system, first create a new dirctory for the 286 C compiler and copy in binaries such as bin/cc, bin/as, bin/ld, bin/strip, bin/nm and libraries such as lib/front, lib/back, lib/comp, lib/small, lib/large. Then excute the *gencc* command in the directory where you are doing the code development. Make sure that you copy 'back' into /lib which allows the 286 C compiler to locate it.

Kernel Debuggers

Kernel debuggers are available for kernel or driver code development. For the MB1 systems, you can choose between a PROM level debugger (DMON) or a kernel debugger. Contact your local Intel sales office to obtain both debuggers. A kernel debugger manual is included in the Appendix. To build kernels with the desired debugger, add module 'db' or 'dmon' in the system file under directory $CONF/systems. For details, refer to the section on "Instructions for Making Kernels".

The kernel debugger is stack-based. Most commands take their operands from the stack and leave results on the stack. Commands are parsed from the command line left to right.

The debugger is entered by pressing the interrupt button. It is also entered if the kernel panics. To leave the debugger, type control-D.

Commands are listed in the following format:

[input_stack] command_name [output_stack] comment

where *input stack* is what the stack should look like before the command is executed, *command name* is the command, and outpur_stack is what will be on the stack after the command is done, and comment is a description of the command.

For example, here is how the addition command (+) would be listed:

[A B] + [A+B] addition

This means that the + command requires two numbers on the stack, which will be denoted by A and B, and that these two numbers on the stack are replaced by their sum.

Here is a listing of debugger commands:

Memory Access Commands

[address] r1 [byte] read 8 bits from address

[address] r2 [word] read 16 bits from address
[address] r4 [long] read 32 bits from address
[data address] w1 write 8 bits to address
[data address] w2 write 16 bits to address
[data address] w4 write 32 bits to address
[address count] dump display count bytes starting at address. Count and address should both be even. Note that in the dump, addresses increase from bottom to top and right to left.

Arithmetic and Logic Commands

[A B] + [A+B]	addition
[A B] - [A-B]	subtraction
	Also available are: *, /, %, <<, >>, <, >, ==, !=, &, |, ↑,
	&&, ||.
	These perform the same operations in the debugger that they do in C.
[A] ! [!A]	not

Stack Manipulation Commands

[A] p [A]	display item on top of stack
[A] pop	remove top item from stack
[A B C ... Z] clrstk	remove all items from stack
[A] dup [A A]	duplicate top of stack

A string of digits is a command to push the number represented by that string onto the stack, and a string of characters that is a kernel symbol is a command to push the value of that symbol onto the stack.

Number Entry

[A] ibase	set input base to A
ibinary	set input base to 2
ioctal	set input base to 8
idecimal	set input base to 10
ihex	set input base to 16
ooctal	set output base to 8
odecimal	set output base to 10
ohex	set output base to 16

Stack Trace

	display a stack trace. Most recently called function is shown first. For ordinary calls, format is name of function, the values of up to three arguments, the values of the first few local variables, where it returns to, and where it's stack frame is. For interrupts, the type and error code are shown, the name of the routine that was called to handle the interrupt, and the registers.

System Dump

sysdump	calls the kernel system dump function. This command may hang or crash the debugger after it does the dump.

Breakpoints

[ADDR] .i brk0	break on instruction at ADDR
[ADDR] .a brk0	break on access of byte at ADDR
[ADDR] .m brk0	break on modify of byte at ADDR
[ADDR] .aw brk0	break on access of word at ADDR
[ADDR] .mw brk0	break on modify of word at ADDR
[ADDR] .al brk0	break on access of long at ADDR
[ADDR] .ml brk0	break on modify of long at ADDR
0 .clr brk0	clear breakpoint in register 0
	The above commands all use breakpoint register 0. You may use brk1, brk2, or brk3 to specify breakpoint registers 1, 2, or 3.
[COUNT] trc0	The next COUNT times that breakpoint 0 is hit, print a short message and continue execution.
[COUNT] trc1	same as above but use breakpoint 1
[COUNT] trc2	same as above but use breakpoint 2
[COUNT] trc3	same as above but use breakpoint 3
db?	show status of all four breakpoints

Other Commands

[A] findsym	prints name of nearest kernel symbol whose value is not greater than A. show inode structure at ADDR
[N] useregs	When a "stack" command is executed, each register set shown is given a number, which is shown in the "set:" field of the stack trace. "useregs" selects set N as the current set.
%eax [EAX]	pushes value of register %eax from the current register set, as defined by "useregs" and the last "stack" command.
%ecx [ECX]	same as above, but reg. %ecx Also available are %edx, %ebx, %esp, %ebp, %esi, %edi, %ds, %es, %fs, %gs, %cs, and %ip. %efl is the flags.
.trap [TRAP#]	pushes trap number of trap that caused interrupt that caused current register set to be saved.
%err [ERR]	pushes error number of trap that caused interrupt that caused current register set to be saved
help	prints a short summary of the dump, brk, trc, db?, stack, and findsym commands.

Software Notes

This section offers some additional information about UNIX System V Release 3.0. Notes about commands, system calls, or files are listed alphabetically and are organized by the Reference Manual where those commands, system calls, or files appear. For example, the section "User Commands" contains notes about commands that are listed in the *User's Reference Manual.* Any further problems may be resolved through the Customer Support Center according to the terms of your maintenance contract.

User Commands

bc

When you enter the following command line and **bc** cannot open *file2*, the error message displayed says that **bc** cannot open *file1*:

bc *file1 file2*

Also, when you enter the following command line and **bc** cannot open *file*, the error message displayed contains garbled characters instead of the name of **file**:

bc *file*

bc

bc does not handle the following two constructs in the same way:

```
(1)   if ( expr ) {
      ...
      }

(2)   if ( expr )
      {
      ...
      }
```

The first case produces what one would expect. The second case is equivalent to an **if** followed by an empty statement, and the compound statement always is executed. If nothing else, the second case should produce a syntax error, but it does not. It dumps core silently.

cdc

The **cdc**(1) command ends abnormally when you invoke it without the **-m** option on an SCCS file which does not have the **v** flag set.

cpio

When you run a series of **cpio** processes in the background, several of them may stop. This occurs because **cpio** creates many child processes, and the maximum number of processes for a user may be exceeded.

You should avoid running too many **cpio** processes in the background.

cpio

The **cpio** command does not properly detect the end-of-tape on the Multibus I port. As a result, multi-volume cpio tape files do not work.

ct

The **ct** command is not compatible with the Basic Networking Utilities. Do not use this command.

cu

The **call** option under **sysadm uucpmgmt** leaves two processes running. To correct this, hit <BREAK> after exiting the menu.

cu

When you use the **cu** command over a direct line, you should enter a carriage return to get a `login` prompt. After the `login` prompt appears, wait about 5 seconds before entering your login. If you do not wait, the first character that you type is not displayed on your terminal screen even though it is accepted by the computer.

This only happens when you use **uugetty**. This does not happen when you use STARLAN network.

cu

The first invocation of **cu** after a powerup may fail; the error message `cannot access device` is displayed. Later invocations of **cu** succeed.

cu

If you do a **cu** from machine_A to machine_B and then do a **cu** from machine_B to machine_C, the command **~%take** *file* will *not* transfer *file* to machine_A.

Transfer files over one link at a time by one of two methods.

- Log in to machine_A, **cu** to machine_B and then to machine_C as described above. The command **~~%take** *file* transfers *file* to machine_B. Then type **~~.** (tilde tilde dot). From machine_B, use the command **~~%take** *file* to transfer *file* to machine_A.
- If possible, **cu** from machine_A to machine_C. You may successfully transfer files over a single link.

cu

Occasionally, **cu** fails with the following message even though all devices are available:

```
NO DEVICES AVAILABLE
```

date

For the AT386, **date** reads only the current date and time from the real time clock, but does not write the clock. The real time clock can be set through the AT CMOS setup program.

file

When you **cc -c main.c**, **file main.o** does not produce the correct output. **file** interprets the optional header, which is not present in the **.o** file.

file

The **file**(1) command reports that a file containing packed data is "data" instead of "packed data."

help

The UNIX system **help** facility is incomplete. Also, the commands of the **help** facility ignore command line options.

help

The commands of the **help** facility ignore SIGHUP, and stay attached to a port that has been dropped or disconnected. This inattentiveness to SIGHUP makes the port useless; you cannot log in because there are two processes reading the port (help's and login's).

Exit **help**.

ipcs

ipcs(1) always reports the number of processes attached to shared memory segments, NATTCH, as zero, even when running processes are currently attached to shared memory segments.

This information in the NATTCH column is incorrect. To obtain the correct information, follow these steps:

Step 1 Invoke **crash** as **root**.

Step 2 Type **od shminfo 3**. The last word on the line will be the number (in hexadecimal of shared memory identifiers).

Step 3 Now type **od shmem 6**.

Step 4 Examine the first **shmem** identifier. If the left-most digit of the third word is in the range 8 to f (hexadecimal) the identifier is in use, and you can get the address for the next step in the last word on the second line. If the left-most digit is not 8-f, skip to Step 6.

Step 5 Type **od** *address* **3**. The left 4 digits of the last word is a hexadecimal number for the number of processes attached to this identifier.

Step 6 To go to the next identifier, add (hexadecimal) 30 to the address reported in response to **od shmem 6** and enter **od** *address* **6**.

Step 7 Repeat Steps 4-6 until all identifiers in use have been displayed. (The data reported by Step 6 will be all zeros.)

login

Intermittently, after executing **exec login**, the following warning message is produced:

```
no utmp entry ... execute from the lowest level shell
```

If the **who** command shows that someone is still logged in on that line, then you must kill the **getty** and allow it to respawn. Otherwise no action is required.

You can avoid the problem by executing **exec su -** *xxxx* (where *xxxx* stands for a login), or by logging out (that is, hanging up or executing an **exit** command) and calling back in.

lp

If you issue the **lp** command on *file* within a directory that has 700 permissions, the following error messages are displayed:

```
lp: can't access file file
lp: request not accepted
```

cat or **pr** *file* and pipe the output to **lp**.

mailx

Interrupts are not handled properly when processing **Cc:**. If interrupts are ignored, nothing happens if an interrupt is received during the composition of the message. However if there is an interrupt during the **Cc:** portion, the **mailx** process dies without saving the letter in **dead.letter**.

od

If a file has an odd number of bytes, **od -c** reports a trailing null byte.

passwd

When you try to change the password for a user that does not exist, **passwd** returns the following error message:

```
Permission denied.
```

This incorrect message appears even if you execute **passwd** as **root**.

ps

When the system is under load, **ps** sometimes reports the following error message:

```
ps: l_lseek() error on lseek, Invalid argument
```

sar

The performance package does not report usage activity on devices which use the i214 driver or any other drivers that do not collect performance statistics. Accordingly, **sar** and any related functions will not report activity on these devices.

sar

sar produces a table in which CPU time is represented as user, system, waiting for I/O, or idle. This idle figure may not be accurate. If memory is being used extensively, the idle time reflected by the output may not be totally idle. Part of the time could be attributed to waiting for memory.

sar

When the command **sar -v** is executed, the field "fhdr-sz" still is reported, even though it should not be.

Ignore this obsolete field that lists all zeros.

sar

The **sar -d** command does not report any information on the Cartridge Tape Controller Subsystem.

sdiff

When doing an **sdiff**, if one of the files contains more than 196 characters in one line, the **sdiff** output loses the separator symbol (|) or loops infinitely, or both.

Do not **sdiff** a file that contains more than 196 characters in a single line.

sh

When the command **sh -c** *string* is executed, the flag value returned is blank. The correct value is **flag=s**. If the **-c** option and the *string* are piped into the shell command, the correct value is returned.

Do not try to execute the **sh** command using the **-t** and **-c** options together. It does not work.

sh

If the path name of the current directory is close to 512 bytes long, then the shell can overwrite internal data structures and cause inconsistencies.

sh

When the system is under a heavy load, you may be logged off if you hit the <BREAK> key several times in succession.

shl

If you hang up while in shell layers, intermittently **/etc/utmp** is not cleaned up (that is, the **who** command still shows that you are logged in, and the **ps** command shows a **getty** running on that line). If someone else then calls into that line, the **login** fails because the **utmp** entry cannot be found. To kill the **getty** corresponding to the affected line, follow this procedure:

Step 1 Execute **ps -ef** to find the process I D of the **getty** running on the line erroneously reported as occupied.

Step 2 Execute **kill -9** *PID* to kill the **getty**. When the **getty** automatically respawns, the problem is cleared.

shl

When using the **shl** command, trapping a "hang-up" signal inside a layered shell suspends the layer. This renders the device unusable until the shell is killed. You receive a warning message that a layer is still running, and then you are returned to the UNIX system shell.

If your terminal does not accept input commands because of the suspended layer, you have to log in at another terminal. To kill the suspended shell, follow this procedure:

Step 1 Find the process number of the shell by executing the following command:

```
ps -eaf
```

Step 2 Look at the output of the command for the appropriate tty (for example, sxt001) and for the process number of the shell.

Step 3 Enter the following command:

kill -9 *PID*

where *PID* is the process number of the shell that you want to kill.

shl

If you are at the console in **shl** and you execute the following sequence of commands, the shell layer does not respond properly:

1. an **stty** command with a carriage return select style argument (for example, **stty cr3**)
2. an **echo** command
3. an **stty** command with a carriage return select style argument (for example, **stty cr0**)

In particular, the layer's prompt does not return until the <BREAK> or <DEL> key is hit. Then, once the prompt appears, the output of any command executed does not print fully or at all until one or more carriage returns are entered.

Return to the **shl** control layer and use the **shl** delete command to delete the layer.

shl

If you execute a background process not in a shell layer that sends output to your terminal, and then you enter shell layers [**shl**(1)], some screen output from the original process may be lost. When you enter **shl**, the output from the original process temporarily stops printing on the screen. The screen output resumes when you exit **shl**(1). Loss of data, if it occurs, is noticeable when the screen output resumes.

This is not normal use of the system and should be avoided.

shl

When resuming a layer of **shl**, the `resuming` *xyz* prompt is often garbled.

tput

If the value of the TERM environment variable is an extremely long string (that is, a string of over 500 characters), **tput**(1) dumps core. Known terminal names [that is, in the **terminfo**(4) data base] are, in general, between 2 and 14 characters.

uname

If you use **uname -S** to change the nodename, and then re-boot the system after a shutdown, the nodename may change to what it was before you used **uname**.

Because the node name may be set with **/etc/rc.d/nodename**, you enter multiuser mode and use **sysadm nodename** to change the system's nodename.

uucp

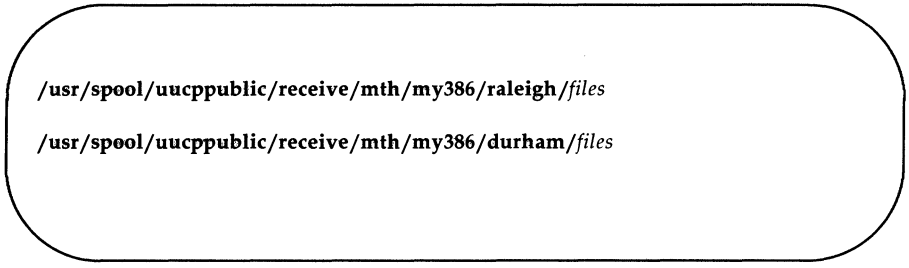
The date stamp included in the mail message by **uucp** to tell you of the arrival of files is always in EST, even when the time zone is set differently on the transmitting and receiving machines.

uulog

If you specify the **f** or **number** option with **uulog** [see **uucp**(1C)], you do not receive a prompt after the execution of **uulog**. To regain access to your terminal, depress the <BREAK> key.

uuto

uuto(1C) does not preserve a directory structure when you try to transfer a directory. For example, suppose that you have two directories, **raleigh** and **durham**, that you want to transfer to user **mth** on a computer named **eagle**. Using two command lines, one for each directory, **uuto** should place the entire contents of the directories on **eagle**, as this example shows:



where **mth** is the destination user and **my386** is the name of your computer. Instead, **uuto** places all *files* from both directories under the following directory:

/usr/spool/uucppublic/receive/mth/my386/*files*

The fix requires adding one line to the **/usr/bin/uuto** program. After the current line 5 in the shell script, add the following line:

```
export UUP
```

The **uuto** code should now look like this:

```
sub=""  (Line 5)
export UUP
mysys='uuname -l'
```

vi

When you use **vi -x** and the terminal shuts off during an editing session, the contents of the buffer are not saved. Contents are saved if the terminal shuts off during an editing session with **vi** without the **-x** option.

vi

The following command line does nothing:

$ vi + *linenumber*

vi

Using a named buffer twice in a **vi** map sequence results in the following error message:

```
Can't put partial line inside macro
```

Use named buffers only once in map sequences.

/usr/news

The directory **/usr/news** is owned by **bin**, belongs to group **bin**, and has permissions 777. Anyone can create and delete news.

Programmer Commands, System Calls

ctime

ctime(3C) reports time in Eastern Daylight Time when the environment variable **TZ** is not set. This may be inconvenient to users outside the Eastern Time Zone.

ctrace

The file produced by running **ctrace** on a small program produces 2 warnings about 'illegal pointer combination, op='.

curses

The **curses**(3X) routines **overlay**() and **overwrite**() incorrectly overlay the source window, *srcwin*, on top of the destination window, *dstwin*. Instead of copying only the text where the two windows overlap as they should, these two routines incorrectly copy the entire source window onto the destination window.

If the source and destination windows are of equal size and equal position on the screen, this incorrect behavior is not apparent to you (that is, the correct and incorrect behavior produce the same results).

curses

Creating a sub-window, using **subwin**(), within a sub-window causes a core dump.

cxref

Running **cxref** on **/usr/src/cmd/pg/pg.c** will cause the following error:

```
SWITCH ERROR IN TMPSCAN:    inafunc = no
```

dial, undial

The **dial**(3C) and **undial**(3C) routines do not work with UNIX System V Release 3.0 **uucp**(1).

fork

The limit on the number of attached shared memory segments per process is not cleared on an **exec**, causing a newly **exec**'ed image to be limited to less than the proper amount of segments.

Set the tunable parameter **SHMSEG** to 45.

fork

When no unused entries remain in the system process table, **fork**() system calls fail. No message is printed on the console to show that the system process table is full. When the **fork**() calls fail, **errno** is set to **EAGAIN**.

terminfo

The **terminfo**(4) entry for the AT&T 4426 terminal contains an incorrect definition for the clear-all-tab-stops (**tbc**) capability, rendering an attempt to use this capability ineffective.

The correct definition for the clear-all-tab-stops capability for this terminal is as follows:

<ESC>F<ESC>[3g

unlink

The **unlink** command fails without issuing an error message when you use it to unlink a busy text file.

System Administrator Commands

chroot

The error message produced when a non-super-user executes the **chroot** command is inaccurate. For example, **chroot /usr/bin** *command* produces the following error message:

```
/usr/bin: Not owner
```

This error message should be interpreted as "chroot: not super-user."

crash

The trace option in **crash**(1M) truncates function names to 8 characters.

crash

When you ask for **help** within **crash** to obtain usage information about a function, it provides the definition of a table-entry or the start-address format even though the function does not require it.

cron

The following message means that during shutdown, **cron** processes are killed:

```
cron aborted: SIGTERM
```

This message is normal.

cron

If you install the Basic Networking Utilities package, the changes made to the **root** crontab (**/usr/spool/cron/crontabs/root**) are not picked up by **cron**.

To correct this, enter the following commands as **root**:

```
# cd /tmp
# crontab -l > ctemp
# crontab ctemp
# rm ctemp
```

dcopy

The usage message issued by **dcopy** does not list all options appropriately. When summarizing, **dcopy** prints the new gap and cylinder sizes when it says that it is printing the old information. The new sizes are, however, printed correctly.

dcopy

dcopy -f recreates the file system based on the `block:inode` count given; however, the blocks specified are assumed to be logical blocks (physical blocks = 2 x the number specified) and the inode count is assumed to be 16 x the number of inodes requested. Thus, **dcopy** does not accept the inode or block count specified to be the actual number of inodes or physical blocks required, as other commands do (for example, **mkfs**).

dd

There are two times when **dd** unjustifiably claims success. The first time occurs when it tries to read from an inaccessible address on a disk. A successful return code is returned with the following message:

```
0+0 records in
0+0 records out
```

The second time occurs when writing to a write-protected disk and diskette. For each block in the file to be copied, the error message that it cannot write to a write-protected disk is displayed, and then the following message is produced:

```
?+1 records in
?+1 records out
```

ff

The **ff** command lists the options **U** and **S** in the usage message. These options are illegal.

finc

The following **finc** error message may be displayed during a backup to cartridge tape:

```
Error occurred, error #=??
```

The error number is echoed to the console. No further information is provided. Check the console for the error number and look it up in the *System Administrator's Guide*.

fsck

The **fsck**(1M) manual page says that when the medium (for example, a floppy disk) is write-protected, **fsck** assumes a **no** response to all questions asked by the command (equivalent to **fsck -n**). This is not true; in these circumstances, **fsck** waits for you to respond to the questions asked.

fsck

If you use **lseek** to skip over entire blocks of a file being written, the kernel assigns a block number of 0 to the skipped blocks. Reads of any bytes in those blocks correctly return a value of 0. However, because the number of "non-zeroed" blocks allocated to the file is consistent with the length of the file, a **fsck** of the file system produces the following message:

```
POSSIBLE FILE SIZE ERROR
```

Core dumps sometimes occur in these circumstances.

There is no damage to the file system. If the file size error messages are bothersome, you can eliminate them by determining the name of the file corresponding to the inode that has the "possible error" with **ncheck**(1M) and, after mounting the file system, copying that file to a temporary file and then back to its original name. Blocks are allocated on the copy to hold all the bytes with values of 0.

fsck

If **/tmp** is filled up with enough file names to run the **root** file system out of inodes, an **fsck** of the file system may produce a POSSIBLE DIR SIZE ERROR message. The file system itself is not damaged.

If the error messages are bothersome, you may eliminate them by running **init 1**, and then moving the contents of the directory that provokes the message to a temporary place, removing and recreating the directory, and then moving the files back. For example,

```
# cd /
# mv tmp badtmp
# mkdir tmp
# find badtmp -print | cpio -pdv tmp
# rm -rf badtmp
# init 2
```

Do not change directory modes and permissions, or the modes and permissions of the files in that directory.

fsck

Due to a difference in physical block size, **fsck** (and **mkfs**) can only be used on block devices (**/dev/dsk/***) on the Multibus I port. Using the raw device (**/dev/rdsk/***) will result in an error message. Either device may be used on the AT386 version.

init

init S drops the remote line and changes speed to 9600 baud when entered from a remote terminal. Only use **init** from the console.

init

Starting in **init**(1M) state 3 and going to **init** 2 changes the modes of the console so that backspace is no longer an erase character. Going from state 2 to state 3 apparently does the same thing.

Restore the original erase *character* with the following command:

stty erase *character*

init

Executing **init s** from within *shl* layers causes inconsistent results. Sometimes, the machine may hang after printing the following message:

```
init: single user mode
```

Other times, the system may not really change run states. At no time within **shl** layers does **init s** do what it is supposed to do.

init

When you go to **init 3**, the listener is already active; thus, the following message appears on the console:

```
/usr/bin/nlsadmin: listener already active
```

Ignore this message.

On the other hand, when you shut the system down, the listener is killed in more than one spot; thus, a message goes to the console that the listener is not running the second time.

labelit

labelit(1M) assumes that it is dealing with a disk file system unless the device path name starts with **/dev/mt** or **/dev/rmt**, which are tape special files for use with cartridge tapes.

mkfs

It is possible to build a file system with **mkfs** (using default inode number) larger than the size of the floppy. It is also possible to create a file as large as 1464 blocks on this file system. A later **fsck** does not complain about either anomaly. It is only when you have to read the file that the read fails.

Create file systems on floppies with **sysadm makefsys** instead of with **mkfs**.

mountfsys

sysadm does not mount an unlabeled file system. File systems that are created by **sysadm** are labeled.

You must label the file system with **labelit**(1M) or use **/etc/mount** manually instead of through the System Administration menus.

shutdown

You cannot use **shutdown**(1M) to return to **init** state 2 from state 3. If you try, you get the following error message:

```
shutdown: Initstate is not for system shutdown
```

shutdown

Sometimes when you shutdown to single-user mode, the unmount of the **/usr** file system fails with the busy error. Manually unmount **/usr**.

sysadm

If the **SIGQUIT** signal (ctrl-¦) occurs while the user is in **sysadm**, a core dump occurs. The core file is in **/usr/admin/menu/*mgmt** (for example, **diskmgmt**, **filemgmt**, as appropriate) and should be removed.

sysadm

If a file system on a floppy is mounted on top of a non-empty directory (for example, **/lib**) with **sysadm**, and then unmounted with **sysadm**, the mount directory becomes inaccessible to all other users. A file system mounted over **/etc** cannot be unmounted since the **umount**(1M) command is no longer accessible.

Avoid mounting a file system on a non-empty directory. If you must mount a file system on non-empty directory, use the "manual" commands such as **mkfs**(1M), **labelit**(1M), **mount**(1M), and **umount** instead of the subcommands in the System Administration menus. If you must use **sysadm**, then you must change the modes of the mount point directory manually with **chmod**(1) after the file system is unmounted.

sysadm

System administration partial subcommand names (for example, **for** for **format**) are not accepted from the shell. Do not use partial subcommand names.

sysadm

The **sysadm** help file for setting a password does not warn of the 8-character limit on passwords. Someone who chooses a password "abcdefgh" is told that the password requires a numeric character but the password is rejected when "abcdefgh9" is chosen because the number does not appear within the first 8 characters.

sysadm

There is no protection provided against multiple users using the same **sysadm** subcommand at the same time.

If you use **sysadm** for floppy disk or for cartridge tape handling, then you must be certain you have control of the appropriate drives. This is no different from using tape drives on larger UNIX system machines. If the session involves changing administrative files, the problem is probably one of system management; specifically, only one person should be authorized to make changes to **uucp** and **passwd** files.

sysadm backup

During a **sysadm backup**, if a bad sector is encountered while writing on a floppy disk, you are not warned of the problem. Furthermore, when a **sysadm restore** is executed, the data from the point of the bad sector throughout the remaining backup disks is unreadable.

You should make frequent individual file or directory backups, with periodic complete backups. It is strongly recommended that you verify a file or a directory that you backup. The file and directory storage commands are available in the **store** menu under the **filemgmt** menu of System Administration.

sysadm portmgmt delete

After executing **sysadm portmgmt delete**, the **inittab** entry for the port is not returned to a usable state. For example, if you connect the modem to tty21 and you execute **portmgmt delete**, the **/etc/inittab** file entry for tty21 looks similar to the following line:

```
21:2:respawn:/etc/getty -t 60 tty21 1200H
```

To change the entry to a usable state, log in as **root**, edit the **/etc/inittab** file, and change the entry for tty21 to look like the following line:

```
21:2:off:/etc/getty tty21 1200
```

sysadm portmgmt modify

If you are connecting a modem to a port that had a terminal connected to it, **sysadm portmgmt modify** may not start the **uugettys**. Before connecting a modem to the port, you should check to see if a **getty** is running on the port. Execute the following command and look for the process number of the port to which you want to connect the modem. You can identify the port by its tty number (for example: tty14 or tty22).

ps -eaf

After you have identified the process number, execute the following command.

kill -9 *PID*

where *PID* is the process number. You may now connect a modem to the port. The **portmgmt modify** command should execute properly.

sysadm uucpmgmt

If you add a dialer script (that is, instructions on how to talk with a modem, ACU, or special device) to **/usr/lib/uucp/Dialers**, then you must also edit the **/usr/lib/uucp/Devices** file to create an entry to use that dialer script. The dialer script is not accessible with **sysadm**.

See the "Supporting Data Base" section in the Basic Networking chapter of the *System Administrator's Guide*.

sysadm uucpmgmt

Using the **sysadm uucpmgmt** command, the subcommand **systemmgmt** should list all systems known to the current machine. If this list contains more than 825 systems (that is, around 5000 characters), the list begins normally, but eventually the output is garbled and the listing ends, with an error message.

sysadm uucpmgmt

sysadm uucpmgmt and **sysadm devicemgmt** use the last two digits of the **tty** name for the I D field in the **/etc/inittab** file. Thus, the same I D field is created for tty11 and tty111, causing **init** to have problems. This problem occurs only when you have 10 or more ports boards.

sysadm uucpmgmt

sysadm uucpmgmt states that you can depress the <CR> key to select the default speed (contty). If you depress the <CR> key, an error message is returned saying that the default speed is not found in the **gettydefs** file. Instead of selecting the default speed, you need to enter a baud rate, for example, 300, 1200, or 9600.

sysadm uucpmgmt

The call option under **sysadm uucpmgmt** leaves two processes running. Hit <BREAK> after exiting the menu.

swap

If there is a heavy process load on a system with a small memory capacity, the **swap** command may silently fail (that is, complete without any error indication) when attempting to add a swap area.

Retry the command when the load on the system decreases. If this is a recurring problem, try to reduce the load on the system or increase the amount of memory on the machine.

swap

The **-d** option does not always work. If there are blocks in use on the swap device, an attempt to delete it will not succeed, and **swap -l** will always show the "INDEL" message.

umount

If you attempt to **umount -d** a disk file system, the error message returned is as follows:

```
umount: /dev/dsk/c1dxxx not mounted
```

The **-d** option is valid only for remote **umount**s.

uucheck

You should always use the **-v** option with **uucheck**. You cannot ask for different levels of debugging information with **uucheck -x**.

uucheck, uucleanup, Uutry

Most of the Basic Networking Utilities commands can be executed from a normal user login. The exceptions are **uucheck** and **uucleanup**, which require either an administrative (**uucp**) login or a **root** login.

uucheck, **uucleanup**, and **Uutry** are located in the **/usr/lib/uucp** directory, which is not in the search path for most logins, including those for **uucp** or **root**. Therefore, you must give the full path name, or you must be in the **/usr/lib/uucp** directory to execute these three commands.

Another alternative is to link the command where it may be easily accessed, for example, **/usr/bin**.

uucico

It is possible to get **uucico** into a runaway state when you use it through the STARLAN network under extremely heavy network load. For example, the process may accumulate too much time (2500 minutes of CPU time).

For example, suppose on machine_A you have a **uucico** to machine_B with suspiciously high CPU time. Log in on machine_B and do **ps -ef**. Look in the output of the **ps** command for a **uucico** process talking to machine_A (that is, with a command line argument like `-s machine_A`). If there is such a process, then the connection is still active. If there is not, then the **uucico** on machine_A is in a runaway state.

If **uucico** is in a runaway state, take the following steps:

Step 1 Kill the **uucico** process with **kill -9** *pid*, where *pid* is the process ID of the looping **uucico**.

Step 2 Remove the associated lock file with **rm /usr/spool/locks/LCK**..*machine* where *machine* is the system where the **uucico** originated, for example, machine_A.

Uutry

There are two legal options to **Uutry**(1M): **-r** and **-x**. Any other input (for example, illegal options, garbage text) is assumed to be a system name.

Consult the **Uutry** manual page in the *System Administrator's Reference Manual*.

volcopy

While executing **volcopy**(1M), you have the option of hitting **DEL** to obtain a shell. On exiting the shell, **volcopy** fails, dumps core, and prints the error message: **bus error- core dumped**.

xts, xtt

The **xts**(1M) and **xtt**(1M) commands will fail if the standard input is not attached to an **xt**(7) channel [that is, not invoked under **layers**(1)], as documented, but an exit status of zero, instead of one, is returned and no error message is printed.

Miscellany

console

If you are running **shl** on the console, and you then run **shutdown -is**, the console hangs.

Do not use **shl** on the console. If you do, stop layers before doing a **shutdown**.

DMON

The kernel cannot be configured with both **DMON** and Weitek floating point support. Either can be used separately.

NTTY driver

Messages from the **NTTY** driver are duplicated on the console.

Kernel

When you run programs that use all available memory and swap space, the computer prints the following message on the console:

```
DANGER: out of swap space
```

However, the programs continue to run.

If a system call fails because a process would exhaust memory or swap space, system error messages (for example, `growreg: cannot allocate ...`) go only to the console. When a system call fails because of lack of memory, the system call always returns an error code and an **errno** value of **EAGAIN**. If these return codes are not checked by your program, you get a core dump when the program tries to use memory.

Problems such as these likely result from programs that have large virtual memory requirements. System administrators should set **MAXUMEM** to some value under the available memory plus swap space so that such processes can fail because they are too big. That notification goes directly to the user.

Nothing can be done with system administration if an application program is available in object code only, except possibly to allocate more swap space with the **swap**(1M) command. If you have source code, you may put checks for error returns into the code and recompile the program.

Kernel

It is possible to get logged off if several interrupts are transmitted during the execution of a pipe. For example, pressing the <BREAK> key repeatedly during a pipe could cause you to be logged off.

Kernel

When a program executes integer division by zero, the following error message is displayed:

```
floating exception - core dumped
```

This message does not accurately describe the error.

Kernel

Processes spawned by the kernel at boot time (**sched, /etc/init, vhand, bdflush**) have start times **(STIME)** that are the time the system was last brought down, not the time they were spawned.

Kernel

Some core dumps may have possible file size errors reported by **fsck**. The reporting of possible file size errors by **fsck** is only a warning, so these errors can be ignored. To determine whether the possible file size errors reported are resulting from core dumps, execute **ncheck -i** *i-number*, where *i-number* is given in the **fsck** message:

```
POSSIBLE FILE SIZE ERROR I=i-number
```

ncheck will generate the path name of a file from its inode number, *i-number*. See the "How To Check a File System for Consistency" section of the *System Administrator's Guide* and the **ncheck** page in the *System Administrator's Reference Manual* for further details.

Kernel

If the system is reconfigured to adjust the maximum number of open files per process, by changing the value of the tunable parameter NOFILES, the value of NOFILE, a #define in **/usr/include/sys/param.h**, and the value of _NFILE, a #define in **/usr/include/stdio.h**, are unaffected. The value of NOFILE and of _NFILE is always 20, regardless of the value of the tunable parameter NOFILES.

Never use NOFILE from **/usr/include/sys/param.h** or _NFILE from **/usr/include/stdio.h** in a user program; use the NOFILES tunable parameter instead.

Kernel

If the operating system runs out of free **clist**s, all input/output activity from/to terminal ports and the console will cease. No warning message is printed by the system to show that it is out of **clist**s.

Kernel

The value of the SHMALL tunable parameter specifies the maximum number of in-use shared memory segments allowable system-wide. This parameter is not checked by the system [that is, **shmget**(2) does not check this limit].

Manual Page for fs Format Is Incorrect

mount expects **s_magic** field of the given argument's super block to be equal to **FsMAGIC**. If not, **mount** does not mount the file system even if it is a proper one. Every file system is made with **s_magic** set to **FsMAGIC**, and therefore **mount** always works. However, the manual page for the **fs**(4) format gives the impression that even if a file system's **s_magic** is not equal to **FsMAGIC**, it can still be a valid file system.

Tunable Parameters

In the "Tunable Parameters" section of the *System Administrator's Guide*, the maximum message size of the message queue facility (**MSGMAX**) is documented to be 128 kilobytes. You cannot set **MSGMAX** to 128K. The largest value (power of 2) that **MSGMNB** (maximum queue size) may be set to is 65,535 (64K -1) bytes.

Saving Device Files When Backing Up root File System

When you backup the root (/) file system, the device files (**/dev** directory) are not saved as part of the backup. To save the device files, become the super-user, mount a blank formatted floppy that has a file system on it, and enter the following commands:

```
# sysadm mountfsys
# find /dev -print | cpio -pdl /mnt
```

where */mnt* is the directory on which the floppy disk file system is mounted. The **cpio** options are lowercase letters **p, d**, and **l**.

To restore the files, insert the floppy on which the files were saved and enter the following commands:

```
# sysadm mountfsys
.
.
.
# cd /mnt
# find . -print ¦ cpio -pdl /dev
.
.
.
# sysadm umountfsys
```

streams

The **streams** system call, "poll", has a timeout problem when given time-out values greater than 20,000,000.

A file **open()** on a streams device blocks even if the no-wait option is set.

f450 Printer Needs Unsupplied Filters

The f450 printer model needs the filter **/usr/lib/etx**, but this filter does not exist. Also, the f450 printer model needs the filter **/usr/bin/450**. This filter is not provided with UNIX System V Release 3.0. The source for **/usr/bin/450** comes with the Source Code Provision of UNIX System V Release 3.0.

The f450 filter is supplied with the Terminal Filters Utilities. Install these utilities before loading the 450 model to ensure that it works correctly.

Read/Write Permissions for Basic Networking Do Not Work

The read/write permissions for the Basic Networking Utilities do not work correctly. For a system to be able to read a directory, the target machine must also have granted the system write permissions (**/usr/lib/uucp/Permissions**). Suppose that your system has read permissions in root and write permissions in **/usr/tmp**. The following command fails:

```
uux "A!pr A!/etc/inittab > A!/usr/tmp/B.out"
```

However, if your system has read permissions in root and write permissions in **/usr/tmp** and **/etc**, the above command line succeeds.

Converting to getopts by Hand

getoptcvt [see **getopts**(1)] adds about 30 lines of code to a shell script, so you may want to convert scripts by hand instead. Converting by hand probably will make the code cleaner and easier to understand. Also, you do not have to worry about parsing option-arguments that are also options.

Follow these guidelines to convert most scripts that currently use the **getopt(1)** command.

Step 1 Delete the old invocation line, and the **if** statement that checks the exit code.

Step 2 Change the **for** loop to a **while** loop that invokes **getopt(1)**.

Step 3 Change the patterns in the **case** statement from **-option** to single option letters.

Step 4 Delete the case for **--**.

Step 5 Add a case for **?**. This case may be used to print the usage message and exit with a non-zero exit code. Note that the **?** is quoted since it is interpreted for file name expansion.

Step 6 Remove all **shift** commands within the **case** statement.

Step 7 Change **$2** to **$OPTARG** for cases that require an option argument.

Step 8 Add the statement **shift `expr $OPTIND - 1`** after the **while** loop so the remaining arguments may be referenced as before.

Here is an example of a script before and after conversion:

```
# before conversion
set -- 'getopt abo: $*'
if [ $? != 0 ]
then
            echo $USAGE
            exit 2
fi
for i in $*
do
            case $i in
            -a | -b)FLAG=$i; shift;;
            -o)OARG=$2; shift 2;;
            --)shift; break;;
            esac
done
```

```
# after conversion
while getopts abo: i
do
            case $i in
            a | b)FLAG=$i;;
            o)OARG=$OPTARG;;
            ?)echo $USAGE
            exit 2;;
            esac
done
shift 'expr $OPTIND - 1'
```

If you want your script to work on releases before UNIX System V Release 3 (that is, use either **getopts** or **getopt**), convert it as the following example shows:

```
if [ "$OPTIND" = 1 ]
then
        while getopts abo: i
        do
        case $i in
        a | b)FLAG=$i;;
        o)OARG=$OPTARG;;
        ?)echo $USAGE
        exit 2;;
        esac
        done
        shift 'expr $OPTIND - 1'
        echo $*
else
        set -- 'getopt abo: $*'
        if [ $? != 0 ]
        then
        echo $USAGE
        exit 2
        fi
        for i in $*
        do
        case $i in
        -a | -b)FLAG=$i; shift;;
        -o)OARG=$2; shift 2;;
        --)shift; break;;
        esac
        done
        echo $*
fi
```

/usr/lib/uucp/Devices

Comments in the **/usr/lib/uucp/Devices** file of the Basic Networking Utilities package show the protocol subfiled attached to the fifth field of the Devices entry. This is incorrect. The protocol subfield should be attached to the first field of the entry.

The corrected comments are as follows:

```
(line 69)    #networkx,eg devicex - - TLIS \D
(line 81)    #              STARLAN,eg starlan - - TLIS \D
(line 87)    #networkx,eg devicex - - TLIS \D nls
(line 94)    #networkx,eg devicex - - TLI \D nls
```

Compatibility Notes

Introduction

By providing UNIX System V Release 3.0, AT&T is continuing its commitment to moving forward by expanding the capabilities of UNIX System V. During the development of the new features of Release 3.0, a greater level of attention was given to maintaining UNIX System V compatibility than before.

Each time we found a particular implementation of a new feature that would break compatibility with previous releases of UNIX System V, we weighed the cost of not providing a critical new feature, or of reimplementing the feature, against the benefits of compatibility. The result is a release of UNIX System V that maintains a high degree of compatibility with previous releases while providing a large variety of new features.

Nonetheless, UNIX System V Release 3.0 is not completely compatible with previous releases. This section details the particular differences so that you can determine if any changes to your software are needed. Some of these changes arise because instances of incorrect behavior in previous UNIX System V releases have been fixed. Others arise from bringing System V into compliance with several developing standards, in particular the /usr/group, IEEE, and ANSI standards. The remainder are irreconcilable differences resulting from adding new features.

Changes from 286 Release 2.0

Release 3.0 is the first UNIX System V/386 release for the 80386. The sections that follow list the changes that affect compatibility with UNIX System V/286 Release 2.0.

Reading this Section

The following section, "Changes in Standard UNIX System V Features," lists those changes that might affect your software whether it takes advantage of the new features of Release 3.0 or not. The other sections, "Changes Under Shared Libraries" and "Changes Under File System Switch," list the changes that might affect your software when it operates under the new features of Release 3.0. Each of these sections is organized by manual page

section, much like the *User's Reference Manual* and *Programmer's Reference Manual*, as follows:

- Changes in Shell Commands
- Changes in System Calls
- Changes in Library Routines
- Changes in File Formats and Contents
- Changes in the C Compilation System
- Miscellaneous Changes

If you will be using the optional Remote File Sharing feature, you should refer to the *Remote File Sharing Release Notes* for a list of additional differences.

Changes in Standard UNIX System V Features

Changes in Commands

Figure 1 lists the UNIX System V commands affected by changes introduced in Release 3.0. The first column gives the command, while the second column gives the title of the section in these *Notes* that describes the change. These titles are also the titles of the manual pages in the *User's Reference Manual* or the *System Administrator's Reference Manual* that describe the commands. Look in the latter manual if the title is marked with a (1M). The last column lists the names of the packages in which the commands are found.

Command	Look Under
cc(1)	*C Programming Language Utilities*
crash(1M)	*System Administration Utilities*
crypt(1)	*Security Administration Utilities (Domestic Only)*
dirname(1)	*User Environment Utilities*
df(1M)	*Essential Utilities*
ed(1)	*Essential Utilities*
ex(1), **vi**(1)	*Editing Utilities*
file(1)	*Directory and File Management Utilities*
fuser(1M)	*System Administration Utilities*
mail(1)	*Essential Utilities*
pr(1)	*Essential Utilities*
ps(1)	*Essential Utilities*
sh(1)	*Essential Utilities*
ex(1), **vi**(1)	*Editing Utilities*
who(1)	*Essential Utilities*

Figure 1: Summary of Changed Commands

cc(1)

Summary of Change: Functions in C programs that do not explicitly return a value will now return a random value. Previously, such functions would often have a zero return value.

While AT&T never guaranteed the behavior of functions that did not return a value, the value zero was often returned. If you have a program that depends on such behavior, it may now fail.

You should check your C programs to ensure that all functions, other than those declared to be of type **void**, always end with a statement of the form

```
return (X)
```

where X is a value of the type appropriate for the function.

crash(1M)

Summary of Change: There are several new and enhanced **crash** commands.

You should refer to the new **crash** manual page in the *System Administrator's Reference Manual* for the new commands and the new method of using some of the old commands.

crypt(1)

Summary of Change: The **crypt** command now takes options that begin with a dash (-), such as **-k**. These can no longer be given as keys. This change will only affect those who have the Security Administration Utilities package installed on their system.

dirname(1)

Summary of Change: The **dirname** command now properly parses the names // and *//anything/* .

Previous to Release 3.0, the **dirname** command would return a dot (.) if given the argument // or *//anything/*. Now it correctly returns a slash, /.

You should change any shell scripts that rely on the previous incorrect behavior of the **dirname** command to reflect the correct operation.

df(1M)

Summary of Change: The **df** command now exits with a non-zero return code when it encounters an error, where before it would always exit with a zero return code.

In earlier releases shell scripts had to check the output of the command to see if it failed. These scripts will still work correctly, but you may want to simplify them.

ed(1)

Summary of Change: The **ed** command now defaults to using the **/usr/tmp** directory to hold temporary files, instead of **/tmp**.

This change does not apply to the **vi** or **ex** programs.

You do not need to take any special action to use **ed** in single user mode if the **/usr** file system is not mounted. When **ed** cannot put its temporary file in **/usr/tmp**, it tries **/tmp** instead.

ex(1), vi(1)

Summary of Change: The structure of the **/usr/preserve** directory used by **vi** and **ex** has changed.

Instead of saving an editing session as a file directly under the **/usr/preserve** directory, it is saved in a subdirectory with the name of the user whose session is saved. Only the same user can access the content of the subdirectory.

In general you will not be able to recover a **vi** or **ex** session preserved before the upgrade to UNIX System V Release 3.0. All sessions should have been recovered before upgrading.

ex(1), vi(1)

Summary of Change: The **ex** and **vi** commands now exit with a return code equal to the number of errors encountered during the editing session. Before, no specified return code was used if errors were encountered.

Because all return codes must be between 0 and 255, if more than 255 errors are encountered, the return code will not be accurate. If an integral multiple of 256 errors are found, then **ex** and **vi** exit with a zero return code.

ex(1), vi(1)

Summary of Change: The **ex** and **vi** commands no longer set the eighth bit in the characters of the % expansion of the current file name.

When the percent sign, %, is used in a shell escape from **ex** or **vi** via the exclamation mark, **!**, the % is replaced with the name of the file being edited. Previously each character in this replacement had the eighth bit set to 1 to quote it; now it is left alone.

Generally, you can use older versions of the **ex** or **vi** commands on Release 3.0, but you cannot use the percent sign, **%**, in a shell escape via the exclamation mark, **!**, even if the file being edited has no special characters in it.

fuser(1M)

Summary of Change: The **fuser** command now exits with a return code equal to the number of errors encountered during execution. In previous releases a zero return code was returned if no errors were encountered; one was returned otherwise.

This change should not adversely affect programs using the **fuser** command.

Because all return codes must be between 0 and 255, if more than 255 errors are encountered, the return code will not be accurate (see **vi** description above).

mail(1)

Summary of Change: The **mail** command now requires an entry for the group **mail** in the **/etc/group** file.

pr(1)

Summary of Change: The **pr** command now correctly interprets the combined options **-m -k** as an error, where before one option would be ignored.

Shell scripts that took advantage of the earlier fault in the **pr** command must be changed to use the correct option.

ps(1)

Summary of Change: The **ps** command now correctly interprets a non-numeric argument to the **-g** or **-p** options as an error, where previously it was treated as zero.

You should change any shell scripts that rely on the previous incorrect behavior of the **ps** command to reflect the correct operation.

Note that the **-g** option is used to examine the processes belonging to a particular process group leader, not to a particular user group identifier.

sh(1)

Summary of Change: A trailing colon in the shell **PATH** variable will cause the current directory to be included in command searches. Previously, a trailing colon was ignored.

sh(1)

Summary of Change: The **test** and **[...]** commands now use the effective user and group IDs to determine permissible file access, instead of the real IDs as previously.

The only way to invoke the **test** command with different effective and real IDs is to invoke the command, or a shell script containing it, from a compiled program that has the set user (group) ID on execution permission. Otherwise, the effective IDs are the same as the real IDs, and this change will have no effect. If your program must rely on the test operators to behave as they did previously, which is to have **test** use the real user and group IDs, you should change it to use the **setuid**(2) or **setgid**(2) system calls to set the effective ID to the real ID before invoking the **test** command or shell script.

sh(1)

Summary of Change: The UNIX system Shell no longer treats the eighth bit in the characters of a command line argument specially; it also no longer strips the eighth bit from the characters of an error message.

If you have any program that sets the eighth bit of characters, they will have to be changed. You should use one of the standard Shell quoting mechanisms, such as the backslash, instead of setting the eighth bit.

sh(1)

Summary of Change: The UNIX system Shell **type** command will display backslashes before each character that was quoted initially.

This change is a result of the change described above, where the eighth bit is no longer used by the shell to quote characters.

sh(1)

Summary of Change: The result of a parameter substitution in a command like

ls "*${a:=xyz abc} lmnop*"

is now correct.

In general, the parameter substitution

${ parameter:=word }

when used inside double quotes and when the *word* contains spaces, now works correctly. If you have programs that rely on the previous incorrect behavior, you should change them to reflect the correct behavior.

who(1)

Summary of Change: The **who -q** command now lists the login names in space-padded fields of equal size and no longer sorts the entries.

If you have any programs that process the output of the **who -q** command, you should inspect them to see if they will still work with the new form of the output.

Changes in System Calls

The following figure lists the UNIX System V system calls affected by changes introduced in Release 3.0. The first column gives the system call, while the second column gives the title of the section below that describes the change. These titles are also the titles of the manual pages in the *Programmer's Reference Manual* that describe the system calls.

System Call	Look Under
acct	**acct**(2)
brk	**brk**(2)
exec	**exec**(2)
fcntl	**fcntl**(2)
fork	**fork**(2)
mount	**mount**(2)
msgrcv	**msgctl**(2), **msgop**(2)
msgsnd	**msgctl**(2), **msgop**(2)
plock	**plock**(2)
ptrace	**ptrace**(2)
semop	**semctl**(2), **semop**(2)
shmat	**shmop**(2)
shmctl	**shmctl**(2)
signal	**signal**(2)
sysi86	**sysi86**(2)
umount	**umount**(2)
ustat	**ustat**(2)

Figure 2: Summary of Changed System Calls

acct(2)

Summary of Change: The **acct** system call now sets **errno** to **EACCESS** when given an argument that is a directory instead of a file, instead of setting it to **EISDIR** as before.

brk(2)

Summary of Change: The **brk** system call fails, setting **errno** to **EAGAIN**, if allocating memory may cause a deadlock.

In previous releases it was possible for the UNIX system to enter a deadlock if free swap space or free main memory were not available. When this occurred, you could not "swap in" a runnable process because there was no room in main memory, and it was also impossible to "swap out" a process to free the needed main memory because there was no room in the swap area.

While rare, this situation occurred often enough to require us to add checks in Release 3.0 to prevent it.

Several system calls now fail and set **errno** to indicate that a deadlock might have occurred. The figure below shows which system calls have changed and what value is given to **errno**.

System Call	errno
brk	**EAGAIN**
exec	**EAGAIN**
fork	**EAGAIN**
plock	**EAGAIN**
shmat	**ENOMEM**
shmctl	**ENOMEM**

Figure 3: Deadlock Detecting System Calls

Note that this new behavior should only occur when the system has nearly exhausted its memory capacity. If this occurs often, you should consider adding more main memory or increasing the size of the swap space.

exec(2)

Summary of Change: The **exec** system call fails, setting **errno** to **EAGAIN**, if allocating memory may cause a deadlock.

See discussion of deadlock detection above.

exec(2)

Summary of Change: The **exec** system call no longer checks the **F_EXEC** bit in the **a.out** file header flags of a program before attempting to execute the program.

This change only affects those programs that were produced using the **-r** option in the **ld** program (also available through the **cc** program.) Such programs are often still executable, so the new behavior allows you to run them. Previously, the **exec** system call would fail with **errno** set to **ENOEXEC**.

fcntl(2)

Summary of Change: The **flock** structure returned with the **F_GETLK** command of the **fcntl** system call has changed; the **l_pid** element has been changed from type **int** to type **short**, and a new element, **l_sysid** of type **short**, has been added.

This change was made to accommodate cases where a file or record lock has been set by a process on a remote computer. The new structure now uniquely identifies a process.

Programs compiled under Release 2.0 that use record locking will have to be recompiled because **l_sysid**, which occupies the space that used to be the high-order 16 bits of the (old Release 2.0-32 bit) pid, can be non-zero for remote locks. This will cause problems if the application tries to kill that pid.

fcntl(2)

Summary of Change: The **fcntl** system call, on a file or record lock request, now sets **errno** to **ENOLCK** when the system runs out of lock resources, instead of setting it to **ENOSPC** or **EMFILE** as before.

This change will only affect programs that currently check for **ENOSPC** or **EMFILE** to learn if the system has run out of lock resources.

Note that the effect will only be seen when you run the program and other programs have used up all the available locks.

fork(2)

Summary of Change: The **fork** system call now has additional reasons for failing, but still sets **errno** to **EAGAIN** in these cases.

This should not affect a program's attempt to catch cases where the **fork** system call fails, because the same **errno** value is used. You should recognize, however, that the new paging system introduces new ways for **fork** to fail when system resources are running low.

mount(2)

Summary of Change: The **mount** system call now correctly fails, with **errno** set to **ENOTDIR**, on an attempt to mount a special file on itself.

mount(2)

Summary of Change: The **mount** system call now fails and sets **errno** to **EINVAL** if the file system's type is not recognized or if the **mflag** (previously **rwflag**) argument is not correct.

plock(2)

Summary of Change: The **plock** system call fails, setting **errno** to **EAGAIN**, if locking a process may result in a memory deadlock.

See discussion of deadlock detection above [**brk**(2)].

ptrace(2)

Summary of Change: The **ptrace** system call now allows write access to a shared text segment. This allows a program to be debugged that more than one person may be running.

In earlier releases, the **ptrace** system call would refuse to write into a text segment, or "pure procedure space," if that segment was being shared with another process and the other process was executing in the segment. The **ptrace** system call allows this in Release 3.0 by ensuring that a separate text image is made and that the write access is to that separate image.

shmop(2)

Summary of Change: The **shmat** system call may fail, setting **errno** to **ENOMEM**, if there is not enough memory to allocate page tables or if attaching the shared segment may cause a deadlock.

Previously the only reason for the **shmat** system call to fail, with **errno** set to **ENOMEM**, was if there was not enough memory for the shared memory segment. Now the unavailability of additional memory for tables needed to manage the separate pages of the shared segment, or the possibility of a memory deadlock, will also cause the system call to fail. [For a discussion of deadlock detection, see the **brk**(2) section above.]

shmop(2)

Summary of Change: **shmat** system call now allows text as well as data segments to be shared.

Previously, a shared memory segment could only be attached to an address in the data segment of a process. With Release 3.0, a shared memory segment can be attached to any address in a process.

shmctl(2)

Summary of Change: The **shmctl** system call now fails, setting **errno** to **ENOMEM** if locking the shared memory region may cause a deadlock.

For example, the following will fail if the attempt to lock the shared memory segment, identified by *shmid*, causes a deadlock:

shmctl(*shmid*, **SHM_LOCK)**

See the discussion of deadlock detection under the **brk**(2) section above.

shmctl(2)

Summary of Change: The **shmctl** system call ignores an attempt to unlock a shared memory segment that is already unlocked, instead of failing as before.

This change is not likely to cause a problem unless a program deliberately tries unlocking a shared memory segment without knowing if the segment is already locked in memory. If such a program looks for the **EINVAL** error return to indicate that the unlock attempt is not needed, it will no longer work as expected.

signal(2)

Summary of Change: The **signal** system call now returns a pointer to a function of type **void** instead of a pointer to a function of type **int** as before.

This change was made to bring System V closer to conforming with the IEEE standard on the UNIX operating system. Since the function to which the return value of the **signal** system call points does not itself return a value, **void** is its correct type, not **int**.

No source code changes are required for Release 3.0, although continued use of the **int** type instead of the **void** type will cause a warning message from the System V **cc** compiler (CPLU 4.0) or the **lint** program checker. While this message is harmless and the code will compile correctly, you should start changing source code now to ensure compatibility with future UNIX systems. All previously compiled programs and application packages that use the **signal** system call will still work with this release.

signal(2)

Summary of Change: The signal **SIGIOT** is being phased out to be replaced with the signal **SIGABRT**.

This change was made to bring System V closer to conforming with the IEEE standard on the UNIX operating system.

Currently, both names are supported so source code is compatible. In the future the name **SIGIOT** will no longer be supported, so you should start changing your source code now. However, the value of **SIGIOT** and the value of **SIGABRT** are the same, which means that all compiled programs, including application packages you may have purchased, will continue to work, even in the future. For example, the **abort**(3C) library routine is now described as issuing the **SIGABRT** signal instead of the **SIGIOT** signal as before. You should therefore write new source code to expect the **SIGABRT** signal. However, since the values are the same, a program previously compiled to expect the **SIGIOT** signal from **abort** will continue to work when linked with the new **abort** routine.

signal(2)

Summary of Change: The **signal** system call now may also fail, setting **errno** to **EINVAL**, if the *func* argument is invalid.

Previously the **signal** system call did not check its second argument, *func*, to ensure that it was one of **SIG_DFL**, **SIG_IGN**, or a valid function address.

sysi86(2)

Summary of Change: The **sysi86** system call, used with the **SI86SWAP** command, now allows the specification of additional swap devices. Previously the system call would fail on second and subsequent attempts.

Release 3.0 now allows more than one swap device. Although it is suggested that the newer **sysi86** command, **SI86SWPI,** be used instead, the less flexible **SI86SWAP** command can still be used, even to specify more than one swap device.

This change should not affect programs, except to eliminate the need to check for the old failure case. We encourage you to change programs to use the new **SI86SWPI** command for future compatibility.

sysi86(2)

Summary of Change: The **sysi86** system call now sets **errno** to **EPERM** for all non-super-user errors.

Previously, when a regular user ran a program that called the **sysi86** system call, if the program used the system call incorrectly, the system call would set **errno** to **EINVAL**. Under the same conditions it now sets **errno** to **EPERM**.

Note that if the super user runs a program that calls **sysi86**, an incorrect use still causes **errno** to be set to EINVAL. Furthermore, this change only affects those **sysi86** uses that require super-user privilege.

umount(2)

Summary of Change: On an attempt to unmount a special device whose major and minor numbers do not exist, the **umount** system call now sets **errno** to **EINVAL**, instead of **ENXIO**.

There should not be many programs affected by this change, since special devices are usually mounted and unmounted using the **mount**(1M) and **umount**(1M) shell commands. You should see if any of your programs that use the **umount**(2) system call check **errno** for the value **ENXIO** when the system call fails. Any that check for **ENXIO** should be changed to check for **EINVAL**.

umount(2)

Summary of Change: For **umount**, **EBUSY** is now returned if the device of the file system to be unmounted is the default pipe device.

The default pipe device is that section of hard disk used for unamed pipes. If it is overridden and placed in a section of disk belonging to a file system that can be unmounted (for example, **/usr**), and you attempt to unmount that file system, the unmount will fail with the above error.

ustat(2)

Summary of Change: A new error return has been added. If the root inode of the mounted file system that you are doing the **ustat** on is NULL, **ENOENT** is set.

Changes in Library Routines

The following figure lists the UNIX System V library routines affected by changes introduced in Release 3.0. The first column gives the routine, while the second column gives the title of the section below that describes the change. These titles are also the titles of the manual pages in the *Programmer's Reference Manual* that describe the routines.

Routine	Look Under
abort	**abort**(3C)
ctime	**ctime**(3C)
fputs	**puts**(3S)
fread	**fread**(3S)
fwrite	**fread**(3S)
gmtime	**ctime**(3C)
localtime	**ctime**(3C)
puts	**puts**(3S)
setvbuf	**setbuf**(3S)
strncat	**string**(3C)
strncmp	**string**(3C)
strncp	**string**(3C)

Figure 4: Summary of Changed Library Routines

abort(3C)

Summary of Change: The **abort** routine now issues the **SIGABRT** signal instead of the **SIGIOT** signal.

See **signal**(2) in the section on system calls.

abort(3C)

Summary of Change: The **abort** routine no longer closes files when the **SIGABRT** (previously **SIGIOT**) signal is being caught or ignored.

Previously the **abort** routine would close all open files before issuing the **SIGIOT** signal that would normally cause the program to halt. If, however, the program had arranged to trap or ignore the **SIGIOT** signal, then it would have to reopen the closed files before continuing.

With Release 3.0 the **abort** routine closes the files only if the program will halt on receiving the **SIGABRT** signal (which has the same value as the **SIGIOT** signal).

If you have a program that used the **abort** routine and trapped or ignored the **SIGIOT** signal, then you should check to see if the new action by **abort** of keeping files open will not cause a problem.

ctime(3C)

Summary of Change: The type of the argument **clock** in the **ctime**, **gmtime**, and **localtime** routines have been changed from "pointer to **long**" to "pointer to **time_t**".

This is another change made to bring System V closer to conforming with the IEEE standard on the UNIX operating system.

No source code changes are required for Release 3.0, but you should start changing source code now to ensure compatibility with future UNIX systems. All previously compiled programs and application packages that use these routines will still work with this release.

puts(3S)

Summary of Change: The **fputs** and **puts** routines now correctly return **EOF** if the attempt fails, instead of zero as before.

If you have a program that checks for a zero return from **puts** or **fputs** to indicate a write error, you should change it to check for **EOF**.

fread(3S)

Summary of Change: The type of the "size" argument to the **fwrite** and **fread** routines and the **strncat**, **strncpy**, and **strncmp** string manipulation routines, has been changed from **int** to **size_t**.

This is another change to bring System V closer to conforming with the IEEE standard on the UNIX operating system. By changing the type to **size_t**, larger buffers or longer strings can be handled by these routines.

No source code changes are required for Release 3.0, but you should start changing source code now to ensure compatibility with future UNIX systems. All previously compiled programs and application packages that use these routines will still work with this release.

setbuf(3S)

Summary of Change: The **setvbuf** routine now behaves correctly as described in the UNIX System V Release 2.0 manual pages.

The correct description of the parameters for the **setvbuf** routine is

```
setvbuf (stream, buf, type, size)
FILE *stream;
char *buf;
int type, size;
```

This is consistent with Release 2.0 and Release 3.0 documentation, but differs from the implementation on UNIX System V Release 2.0. The implementation now correctly matches the documentation.

If you have a program that uses **setvbuf** with the second and third arguments switched to match the earlier implementation, you will need to switch the arguments back to have the program work on Release 3.0.

string(3C)

Refer to the **fread**(3S) section above for the changes made to the **string** routines.

Changes in the C Compilation System

There are three general changes in the C Compilation System utilities, or C Programming Language Utilities as they are now called.

- The recognition of a new C preprocessor directive, **#ident**, and the addition of it to Release 3.0 header files.
- The addition of new error codes and system calls returning new or different error codes.
- Changes to Release 3.0 header files.

Your programs may be affected by these changes.

#ident

> Summary of Change: The new C preprocessor recognizes a new directive, **#ident**, that provides a better way of controlling software versions. This directive has been added to all header files.

The change will only cause a problem if another C preprocessor is used that does not recognize the **#ident** directive, and any of the standard header files are included in the code processed by the other preprocessor. In such cases you will have to arrange for the header files to be stripped of the **#ident** directives before being passed to the other preprocessor.

New Error Codes

Figure 5 lists the error codes introduced in Release 3.0.

Error Code	Value in Release 2.1	Value in Release 3.0
ENOSTR	50	60
ENODATA	51	61
ETIME	52	62
ENOSR	53	63
ENONET		64
ENOPKG		65
EREMOTE		66
ENOLINK	67	67
EADV		68
ESRMNT		69
ECOMM		70
EPROTO		71
EMULTIHOP		74
EBADMSG		77
ELIBACC		83
ELIBBAD		84
ELIBSCN		85
ELIBMAX		86
ELIBEXEC		87

Figure 5: Error Codes

The values of four error codes introduced were changed for Release 3.0. AT&T did not support the four error codes in Release 2.0, and there were no programs or routines generally available that used the codes, so there should be no affect from these changes.

The codes **ENONET** through **ECOMM** and the **EMULTIHOP** code are specific to the Remote File Sharing feature. The **ENOSTR**, **ETIME**, **ENOSR**, **EPROTO**, and **EBADMSG** codes are related to the use of Streams and the Transport Level Interface features. The **ELIBACC** through **ELIBEXEC** codes are related to the use of shared libraries. You should not see any of these error codes unless you are using one of the related features.

Several system calls now include additional error codes as possible reasons for failure. Some of these codes are related to the new features as mentioned above, but some are older error codes that are now appropriate failure reasons in Release 3.0. In some cases, system calls simply have a different error code for the same failure. These changes are described in the various sections under "Changes in System Calls" in this document.

Changes in Header Files

Figure 6 lists the header files changed, dropped, or introduced in Release 3.0. The manual pages for the system calls and routines listed in the *Programmer's Reference Manual* describe where some of these header files are needed. The names listed are those referred to in the manual pages and are the names used in **#include** statements. The files are found starting in the **/usr/include** directory. Some of these files are (or were) only available if you have a source license.

Most of the changes should not affect normal user programs. However, some header files changed greatly as a result of adding new features, for example, paging in Release 3.0. These header files are marked with an asterisk (*) in the figure below. Programs that use these files will, at minimum, have to be recompiled. In some cases, especially if a particular order was assumed in a structure or a particular offset was assumed for an element of a structure, the programs will have to be recoded to remove such dependencies. Such programs are extremely system-dependent and, as such, cannot be expected to be fully portable to new releases of UNIX System V.

Header File	Release 3.0
core.h	changed
errno.h	changed
fcntl.h	changed
filehdr.h	changed
ldfcn.h	changed
limits.h	new
signal.h	changed
stdio.h	changed
string.h	changed
unistd.h	new
sys/buf.h	changed
sys/dir.h	changed
sys/fblk.h	changed
sys/fcntl.h	new
sys/filsys.h	changed
* sys/flock.h	changed
* sys/immu.h	changed
sys/inline.h	new
* sys/inode.h	changed
sys/ipc.h	changed
sys/lo.h	dropped
sys/map.h	changed
sys/mau.h	changed
* sys/mount.h	changed
sys/open.h	changed
* sys/param.h	changed
* sys/pfdat.h	new
* sys/proc.h	changed
* sys/reg.h	changed

Figure 6: Changed, Dropped, of Added Header Files (page 1 of 2)

Header File	Release 3.0
sys/shm.h	changed
sys/stat.h	changed
* sys/stream.h	new
* sys/stropts.h	new
sys/sysi86.h	changed
* sys/sysinfo.h	changed
sys/sysmacros.h	changed
sys/systm.h	changed
sys/tty.h	changed
sys/types.h	changed
* sys/user.h	changed
* sys/var.h	changed
sys/fs/s5dir.h	new
sys/fs/s5filesys.h	new
sys/fs/s5param.h	new
sys/fs/s5inode.h	new
sys/fs/s5fblk.h	new

Figure 6: Changed, Dropped, of Added Header Files (page 2 of 2)

Because of the extensive changes made to the header files marked with an asterisk (in Figure 6), the changes are not discussed in the sections that follow. Instead you are referred to the files themselves for a description of the changes.

core.h

Summary of Change: The macros defined in **core.h** that specify the location of the stack in a core dump file were changed to reflect the change in the stack location.

errno.h

Summary of Change: Several new error codes have been added for the new features of Releases 2.1 and 3.0. See other parts of this document for details, in particular Figure 5, Error Codes.

fcntl.h

Summary of Change: The **flock** structure missing from the **fcntl.h** file in some Release 2.0 computers is now included. Also, a copy of this file is now in **sys/fcntl.h**.

These changes should not affect any programs. However, we encourage you to start changing your programs to include **sys/fcntl.h** instead of **fcntl.h.**

filehdr.h

Summary of Change: A new macro has been added to **filehdr.h,** corresponding to the new **F_BM32MAU** bit in the **a.out** file header flags of a program.

ldfcn.h

Summary of Change: The declarations of some functions already declared in **stdio.h** have been removed from **ldfcn.h**

Since the proper use of the **ldfcn.h** header file requires the inclusion of the **stdio.h** header file, this change should not affect any programs.

signal.h

Summary of Change: New macros for the Release 3.0 enhanced signal handling have been added. Also, the macro **SIGABRT**, equivalent to **SIGIOT**, has been added. The **SIG_DFL** and **SIG_IGN** macros have been changed from values of type **(int(*)())** to values of type **(void(*)())**.

The latter two changes are discussed under the **signal**(2) section above.

stdio.h

Summary of Change: Several functions that return values of type **int** are now explicitly declared in **stdio.h** rather than leaving them implicitly declared.

Unless a program has already declared or defined symbols with the same name but of different type, this change should not present a problem and should help program developers find improper use of the functions. The functions that are now explicitly declared are listed as follows:

fclose	**fflush**	**fgetc**	**fprintf**
fputc	**fputs**	**fread**	**fscanf**
fseek	**fwrite**	**getw**	**pclose**
printf	**sprintf**	**puts**	**putw**
scanf	**setvbuf**	**sscanf**	**system**
ungetc	**vfprintf**	**vsprintf**	**vprintf**

string.h

Summary of Change: All the definitions found in the **memory.h** header file are now also included in the **string.h** header file.

This change should not cause a problem unless a program includes the **string.h** file but defines its own versions of the symbols from the **memory.h** file. The symbols added to the **string.h** file are **memccpy**, **memchr**, **memcpy**, **memset**, and **memcmp**. These are not macros, so they cannot be undefined. If your program already defines one or more of these symbols, it is best if you redefine them with different names.

The **memory.h** file still exists—programs that refer to it can still be compiled on this release. The change will not affect programs already compiled.

sys/buf.h

Summary of Change: The file system dependent fields, **b_filsys** and **b_dino**, have been removed from the **buf** structure defined in the **sys/buf.h** header file.

sys/dir.h

Summary of Change: The contents of **sys/dir.h** and **sys/fblk.h** have been moved to the header files **sys/fs/s5dir.h** and **sys/fs/s5fblk.h**, respectively. The old files contain a **#include** directive to include the contents from the new files. Also, the type of the **d_ino** field in the structure has changed from **ino_t** to **ushort**.

Because the standard System V file system type is just one of, potentially, several others, the contents of **sys/dir.h** and **sys/fblk.h** were moved to the **sys/fs** directory under different names where there will be similar files for other file system types provided by AT&T in the future. Thus we encourage you to change statements such as:

```
#include <sys/dir.h>
#include <sys/fblk.h>
```

in any C programs you may have to the following statements.

```
#include <sys/fs/s5dir.h>
#include <sys/fs/s5fblk.h>
```

The old **sys/dir.h** and **sys/fblk.h** files may be removed in the future.

The type change should not cause a problem because the header file that defines the **ino_t** type, **sys/types.h**, had to be included previously, and, secondly, the **ino_t** type was and is the same as the **ushort** type (namely **unsigned short.)**

sys/fblk.h

See the **sys/dir.h** section above for information.

sys/fcntl.h

See the **fcntl.h** section above for information.

sys/filsys.h

Summary of Change: The contents of **sys/filsys.h** have been moved to the header file **sys/fs/s5filsys.h**. The old file contains a **#include** directive to include the contents from the new file. Also, the type of the **s_inode** and **s_tinode** elements in the **filsys** structure have changed from **ino_t** to **ushort;** and **getfs** is now declared as a macro instead of a function.

Because the standard System V file system type is just one of, potentially, several others, the contents of **sys/filsys.h** were moved to the **sys/fs** directory under a different name, where there will be similar files for other file system types provided by AT&T in the future.

The type change should not cause a problem because the header file that defines the **ino_t** type, **sys/types.h**, had to be included before, and second, the **ino_t** type was and is the same as the **ushort** type (namely **unsigned short**.) However, the change in **getfs** from a function to a macro requires that you recompile any programs that use it. We encourage you to start changing your programs to include **sys/fs/s5filsys.h** instead of **sys/filsys.h**, as this file may be dropped in a future release.

sys/inline.h

Summary of Change: Some of the macros defined in **sys/inline.h** are not defined when using the **cxref** command to cross-reference a program that includes this file.

sys/inode.h

Summary of Change: The contents of **sys/inode.h** have been changed extensively and now contain only inode information that is file system independent. The inode information that depends on the System V file system is now contained in **sys/fs/s5inode.h**.

Any programs that include **sys/inode.h** will also have to include **sys/fs/s5inode.h**. Because of the extensive changes, it is suggested that you compare the old file with the two new files to determine what coding changes, if any, are required.

sys/ipc.h

Summary of Change: The definitions of the macros **SHM_LOCK** and **SHM_UNLOCK** have been moved from **sys/ipc.h** to **sys/shm.h**.

Since the **SHM_LOCK** and **SHM_UNLOCK** macros are only used in programs that already include the **sys/shm.h** file, this change should not have any affect.

sys/lo.h

Summary of Change: The **lo.h** file has been dropped in Release 3.0.

sys/map.h

Summary of Change: The declarations for **swapmap**, **coremap**, and **shmmap** have been deleted from **sys/map.h** in Release 3.0.

sys/open.h

Summary of Change: A macro for another type of driver-level open, **OTYP_SWP**, has been defined in Release 3.0; it takes the same value as the previous **OTYP_LYR** macro, which now takes on a new value. The macro **OTYPCNT**, which gives the total number of types, has been increased by one.

Any drivers that use **OTYP_LYR** will have to be recompiled.

sys/param.h

Summary of Change: Parts of **sys/param.h** have been changed extensively; most of the changed content has been moved to the header file **sys/fs/s5param.h**. The old file contains a **#include** directive to include the contents from the new file.

You should examine the **sys/param.h** and **sys/fs/s5param.h** files to see if the changes will affect any of your programs. Since the new file is automatically included by the old file, your programs that need the new file's content should not also include the file.

sys/proc.h

Summary of Change: While **sys/proc.h** has changed extensively, the offsets of certain elements in the **proc** structure (the process table) have not. These are described in the *Driver Development Utilities Guide*.

sys/shm.h

Summary of Change: The macro **SHM_CLEAR** has been renamed to **SHM_INIT**, although the value remains the same, and the **shm_reg** element in the **shmid_ds** structure has changed from a segment descriptor structure to a pointer with sufficient padding so as not to affect other elements. Also, the definitions of the macros **SHM_LOCK** and **SHM_UNLOCK** have been moved here from the **ipc.h** header file.

The renaming of **SHM_CLEAR** to **SHM_INIT** will not affect compiled programs or application packages that use this macro because the same value has been kept. However, source code should be changed to reflect the new name.

sys/stat.h

Summary of Change: The type of the **st_ino** element in the **stat** structure has changed from **ino_t** to **ushort**. Also, new definitions have been added to the **stat.h** header file.

The type change should not cause a problem because the header file that defines the **ino_t** type, **sys/types.h**, had to be included before, and, secondly, the **ino_t** type was and is the same as the **ushort** type (namely **unsigned short)**.

The addition of the new definitions should not cause a problem unless a program includes this file, using a C program line such as the following:

```
#include <sys/stat.h>
```

and the program already uses a symbol now defined in the header file.

The new symbols are listed below:

S_ENFMT	**S_IRWXU**	**S_IRUSR**	**S_IWUSR**
S_IXUSR	**S_IRWXG**	**S_IRGRP**	**S_IWGRP**
S_IXGRP	**S_IRWXO**	**S_IROTH**	**S_IWOTH**
S_IXOTH			

sys/sysmacros.h

Summary of Change: The macro **brdev**, that would mask the lower 17 bits of a device number, has been removed from **sys/sysmacros.h**.

sys/systm.h

Summary of Change: The declarations for the variable **Maxmem** and the functions **bmap**, **ialloc**, **owner**, and **maknode** have been removed from **sys/systm.h**.

sys/tty.h

Summary of Change: A declaration for a new variable, **cfreecnt**, of type **int**, has been added to **sys/tty.h**.

sys/user.h

Summary of Change: The **sys/user.h** header file has changed extensively.

sys/fs/s5dir.h
See the **sys/dir.h** section above for more information.

sys/fs/s5filsys.h
See the **sys/filsys.h** section above for more information.

sys/fs/s5param.h
See the **sys/param.h** section above for more information.

sys/fs/s5inode.h
See the **sys/inode.h** section above for information.

sys/fs/s5fblk.h
See the **sys/dir.h** section above for information.

Miscellaneous Changes

Longest Allowed Path Names

Summary of Change: The longest path name is now restricted to 1024 bytes. System calls that require path names as arguments will now fail, setting **errno** to **ENOENT**, if a longer path name is given.

The length of a file's full path name is now restricted to 1024 bytes. While previously the path name was not restricted by the UNIX operating system, most programs gave an ad hoc limit to the length. Generally these limits were well below 1024 bytes, so most programs should not be affected by this change.

The **limits.h** file defines a macro **PATH_MAX** to be the longest length of a path name. While in Release 3.0 this file incorrectly sets the macro to 256, it will probably be changed in a future release to 1024. Local system administrators can safely change the value for **PATH_MAX** to 1024 without harm, since the Release 3.0 system internally uses the longer limit.

We encourage you to include the **limits.h** file with a statement like

```
#include <limits.h>
```

and refer to the **PATH_MAX** macro for the longest path name allowed.

Changes in Commands

chroot(1)

Summary of Change: Using either the **chroot** shell command or the **chroot** system call may not work when the program involved has been built with a shared library.

This will not affect any existing programs. However, new programs compiled in this release, or old programs recompiled in this release, may have a problem if the libraries linked into the program are shared. When such a program is run, the UNIX system tries to find the shared library using the original path name of the library, but if the root has changed places, the system may be unable to find the library.

The only way to avoid this problem when using either the **chroot** command or the **chroot** system call is to arrange beforehand for copies of the required shared libraries to be found in the correct place under the new root directory.

For example, suppose a program named **xyz** has been compiled to use the shared library **/shlib/libc_s**. The following commands will let this program run under a new root directory called **/abc**.

```
mkdir /abc/shlib    # only if /abc/shlib does not exist yet
cp /shlib/libc_s /abc/shlib/libc_s
chroot /abc xyz
```

Changes in System Calls

chroot(2)

Refer to the **chroot**(1) section in previous text for changes to the **chroot**(2) system call.

exec(2)

Summary of Change: The **exec** system call will now fail if the program to be run requires a shared library for which you do not have execute permission (**errno** set to **ELIBACC**), or if you try to exec a shared library directly (**errno** set to **ELIBEXEC**), or if you try to exec a shared library that doesn't exist.

mount(2)

Summary of Change: The **mount** system call now fails and sets **errno** to **EINVAL** if the file system's type is not recognized or if the **mflag** (previously **rwflag**) argument is not correct.

Enhancements to Earlier Releases

Below is a summary of customer-generated modification requests against earlier UNIX system releases that were fixed for Release 3.0 on your computers.

Component	Description
at	at fails in deeply nested directories.
at	at/batch won't access queue "z".
calendar	calendar doesn't clean up temporary file.
cpio	cpio may hang.
cpio	handling end-of-media different from I/O errors.
crash	crash command on live system "freezes" running procs at program entry.
crash	crash core dumps if you try to trace an empty proc slot.
crypt	when used in a pipeline, crypt fails intermittently.
crypt	crypt(1) core dumps when /dev/tty has wrong permissions.
cut	a bug is in cut when backspaces are in text.
cut	cut dumps core when cutting a field that is too long (–f option only).
cut	cut dumps core if given a line without a newline.
dd	dd can write past end of disk using blocked device.
ed	removal of Editing Utilities does not remove everything.

Component	Description
ed	ed reports incorrect line number on "line too long".
ed	the undo command nullifies the effect of the last command without warning the user.
ed	editor won't show last line in file.
ed	ed exits with 0 on failure.
egrep	"egrep -f file -i" dumps core.
find	find -cpio does not work; manual page incomplete.
fuser	fuser always exits with a return code of 1.
help	help term2 doesn't display list of known terminals.
init	control-d is ignored in single-user mode.
kernel	semctl() doesn't complain if semval exceeds maximum on SETALL cmd.
kernel	size of shared memory segment set by first shmget syscall.
kernel	undocumented shared memory action can degrade system.
libcurses	addch() causes core dump.
libcurses	flash does not flash the screen each time called.
libcurses	nodelay generates a hangup.
libcurses	curses needs way to determine if the xt driver is used.
libcurses	libcurses does not work on 5420 terminal.

Component	Description
libcurses	mvscanw() won't compile.
libcurses	tgetstr(id, area) -- parameter "area" ignored by the function.
libcurses	unreferenced symbol in libcurses.a.
libcurses	wscanw doesn't echo chars back.
libcurses	the getstr() function in curses does not echo properly on 3B2.
mail	mail has a pointer problem.
mail	mail silently quits with exit code 0 on garbaged mail file.
mailx	long subject line causes core dump.
mailx	mail is lost when /usr runs out of space.
mailx	mailx "Subject" uses VTIME instead of VMIN.
pg	shell escape has problems when pg is in pipeline.
pg	pg prints garbage when going between files.
pr	pr -m dumps core.
sadmin	ncpio does not work for large inode number.
sadmin	makefsys will accept 0 for the number of inodes.
sh	improper termination of processes when piped reader process dies.
sh	shell does not recognize trailing colon as current directory in PATH.
sh	error in shell test.c code.

Component	Description
shl	the swtch character gets "eaten" by the tty driver.
shutdown	shutdown complains from single user state to firmware state.
tar	tar attempts to open the tape drive when it should create an archive file on disk.
terminfo	request for terminfo de-compiler. (infocomp)
uucico	TCSETA should always be TCSETAW.
uucp	default Systems file should include cuuxb instead of nwuxd.
vi	vi -r -x file destroys saved editing session.
vi	vi <file> -I causes memory fault core dumped.
vi	vi adds characters to EMPTY files.
vi	vi marks file as modified on initial entry.
volcopy	volcopy reports from_tape empty if to_tape has no volcopied label.
volcopy	volcopy.c fslog routine problem.
wall	wall gets confused by layers.
who	A DMD terminal running layers will not show up in a who cmd.

Documentation

The following documentation is included with UNIX System V Release 3. See the *Documentation Roadmap* for a complete list of documentation that supports AT&T products that you may use with UNIX System V Release 3.0 on AT&T computers. For more information about other products, see the *Product Overview* for the product you're interested in.

Document Title	Select Code	Status
AT&T 3B2 Computer UNIX System V Release 3 Documentation Roadmap	305-555	**updated**
UNIX System V Release 3.0 INTEL 80286/80386 Computer Version Product Overview	307-643	**new**
UNIX System V Release 3.0 INTEL 80286/80386 Computer Version Programmer's Reference Manual	307-645	**new**
UNIX System V Release 3.0 INTEL 80386 Computer Version Release Notes	307-646	**new**
UNIX System V Release 3.0 INTEL 80286/80386 Computer Version System Administrator's Guide	307-681	**new**
UNIX System V Release 3.0 INTEL 80286/80386 Computer Version System Administrator's Reference Manual	307-682	**new**
UNIX System V INTEL 80286/80386 Computer Version User's Guide	307-683	**new**
UNIX System V Release 3.0 INTEL 80286/80386 Computer Version User's Reference Manual	307-684	**new**

Additional Documentation

A new *Programmer's Guide* that you may purchase separately consolidates the contents of two previously existing documents (the *Programming Guide* and the *Support Tools Guide*) and adds new material. It describes how to use many of the UNIX system's programming tools by presenting different program development scenarios. It also contains chapters that give details about important UNIX system programming tools, such as new information for the Terminal Information Utilities package and the Shared Libraries feature.

The new *STREAMS Primer*, provided with the optional Networking Support Utilities product, presents an overview of STREAMS for programmers who will be using STREAMS. Two other new networking support documents may also be purchased: the *STREAMS Programmer's Guide*, and the *Network Programmer's Guide*.

The *System Administrator's Guide* has been updated to include information for administering the optional Remote File Sharing feature and new features of **uucp**.

Document Title	Select Code	Status
UNIX System V Release 3.0 INTEL 80286/80386 Computer Version Programmer's Guide	307-644	**new**
UNIX System V Release 3.0 INTEL 80286/80386 Computer Version STREAMS Primer (delivered with the optional Networking Support Utilities product)	307-649	**new**
UNIX System V Release 3.0 INTEL 80286/80386 Computer Version STREAMS Programmer's Guide (delivered with the optional Networking Support Utilities product)	307-680	**new**
UNIX System V Release 3.0 INTEL 80286/80386 Computer Version Network Programmer's Guide	307-641	**new**
UNIX System V Release 3.0 INTEL 80286/80386 Computer Version Networking Support Utilities Release 1.0 Release Notes (delivered with the optional Networking Support Utilities product)	307-642	**new**
UNIX System V Release 3.0 INTEL 80286/80386 Computer Version Remote File Sharing Utilities Release 1.0 Release Notes (delivered with the optional Remote File Sharing Utilities product)	307-647	**new**

Preface

Conventions Used in These Release Notes

In this document, as in all UNIX system documentation, certain typesetting conventions are followed when command names, command line format, files, and directory names are described. There are also conventions for displays of terminal input and output.

- You must type words that are in **bold** font as they appear.
- *Italic* words are variables; you substitute the appropriate values. These values may be file names or they may be data values, as applicable.
- CRT or terminal output and examples of source code are presented in `constant-width` font.
- Characters or words in square brackets, [ ], are optional. (Do not type the brackets.)

A command name followed by a number, for example, **ed**(1), refers to that command's manual page, where the number refers to the section of the manual. Manual pages from section (1) appear in the *User's Reference Manual*, unless otherwise noted. Manual pages from sections (3) and (4) appear in the *Programmer's Reference Manual*. Manual pages from section (1M) appear in the *System Administrator's Reference Manual*.

Examples in these *Release Notes* show the default system prompt for UNIX System V, the dollar sign (**$**). They also show the default prompt when you log in as the super-user, the pound sign (**#**).

These *Release Notes* refer to packages and to products. A package is a group of programs that do related things. For example, the Editing Utilities package contains UNIX System V's text editors and their associated files. UNIX System V Release 3.0 includes many packages. A product is something that you purchase independently of UNIX System V Release 3.0, for example, Remote File Sharing.

Introduction

The Networking Support Utilities (NSU) package is an optional System V Release 3.0 product that supplements the Essential Utilities by extending system capabilities to support networking applications. The product includes software support for STREAMS, the AT&T Transport Interface, and the Listener.

The Networking Support Utilities product is required to take advantage of the following features of Release 3.0: Remote File Sharing product, STREAMS mechanisms and tools, the AT&T Transport Interface, the enhanced Basic Networking Utilities, and the Listener.

STREAMS

STREAMS is a general, flexible facility for developing UNIX communication services. By defining standard interfaces for character input/output within the kernel, STREAMS supports development ranging from complete networking protocol suites to individual device drivers. The standard interfaces and associated tools enable modular, portable development and easy integration of network services and their components—these were used to develop protocol modules and device drivers for Release 3.0. STREAMS provides a broad framework that does not impose any specific network architecture. It implements a user interface consistent and compatible with the character I/O mechanism that is also available in the UNIX system.

The power of STREAMS resides in its modularity. The design reflects the layering characteristics of contemporary networking architectures. Each basic component (called a module) in a STREAMS implementation represents a set of processing functions and communicates with other modules via a standard interface. From the user level, kernel resident modules can be dynamically selected and interconnected to implement any rational processing sequence. No additional kernel programming, assembly, or link editing is required. Modularity allows for the following advantages:

- User-level programs (commands such as **uucp**) are independent of underlying protocols and communications media so the programs need not be changed when new media or protocols between systems become available.

- Network architectures and higher-level protocols are independent of underlying protocols, drivers, and media.
- Higher-level services can be created by selecting and connecting lower-level services and protocols.

In addition to the standard interfaces, STREAMS provides a set of software tools that help source customers build modules and drivers.

Several new documents have been written describing how to use STREAMS. For more information, see the "Documentation" section at the end of these *Release Notes*.

AT&T Transport Interface

With Release 3.0, UNIX System V supports a Transport Interface based on the Transport Service Definition (Level 4) of the International Standards Organization (ISO) Open Systems Interconnection (OSI) reference model. The transport service supports two modes of transfer: connection mode and connectionless mode. Connection mode is circuit-oriented and supports data transfer over an established connection in a reliable, sequenced manner. The connectionless mode is message-oriented (datagrams) and supports data transfer in self-contained units with no logical relationship required among units.

The AT&T Transport Interface defines how a user accesses the services of a transport protocol, called a Transport Provider. An example of a Transport Provider is the Universal Receiver Protocol (URP). Applications programs access the Transport Provider by using the Transport Interface routines in the new Network Services Library. These routines support access to a Transport Provider in a media- and protocol-independent manner. The Transport Provider uses kernel level programs to send the information to the desired physical device, such as the STARLAN Network Access Unit (NAU). By using the AT&T Transport Interface, application programs will be able to access other Transport Providers that may be available in the future.

Listener

The Listener is a program that can be used with Transport Providers on a system. The purpose of the Listener is to receive requests for services from another system, interpret which service is needed, and start a process that has been named to provide the requested service. The Listener then drops out of the communications path and continues to listen for new service requests.

For more information about the Listener, see the *Programmer's Reference Manual* and the *Network Programmer's Guide*.

Contents of the Release

The Network Support Utilities comes either on one floppy diskette or cartridge tape and requires approximately 440 free blocks in **/usr** and 651 free blocks in /.

513 of the blocks in / are for temporary space that is needed to reconfigure the system. After the package is installed, only about 140 blocks are used.

The following files are contained on the diskette:

/etc/conf/modules/clone/config
/etc/conf/modules/clone/clone.o
/etc/conf/modules/log/config
/etc/conf/modules/log/log.o
/etc/conf/modules/log/space.c
/etc/conf/modules/timod/config
/etc/conf/modules/timod/timod.o
/etc/conf/modules/timod/space.c
/etc/conf/modules/tirdwr/config
/etc/conf/modules/tirdwr/tirdwr.o
/etc/conf/modules/tirdwr/space.c
/usr/bin/nlsadmin
/usr/bin/strace
/usr/bin/strerr
/usr/bin/strclean
/usr/include/sys/lihdr.h
/usr/include/sys/tiuser.h
/usr/include/sys/tihdr.h
/usr/include/sys/strlog.h
/usr/include/sys/log.h
/usr/include/listen.h
/usr/include/tiuser.h
/usr/lib/libnsl_s.a
/usr/lib/libnls.a
/usr/net/nls/listen
/usr/options/nsu.name

Installation Procedures

If you received this product as a separate package, and installation is required, this section describes the installation procedures for the Networking Support Utilities.

Prerequisites

Before you can install the Networking Support Utilities you must complete the following prerequisites.

Software

You must have installed the following System V Release 3.0 software before installing the Networking Support Utilities:

- Essential Utilities
- Directory and File Management Utilities

Hardware

Networking Support Utilities can be run on your computer. The minimum hardware configuration required is 2 megabytes of main memory.

Installation Procedure

The following list describes the installation procedures for the Networking Support Utilities. The Networking Support Utilities for your computer are distributed either on one floppy diskette or cartridge tape. Most of the utilities are object code files. To begin the installation:

- Login as the super-user and be sure you are in the root (/) directory.
- Place your computer in system state—2 (multiuser) or 1 (single-user). To run this procedure in single-user mode, you must mount **/usr**.

Run sysadm installpkg

Step 1: To install the Networking Support Utilities, use the direct access method of the System Administration menu as follows:

```
# sysadm installpkg
```

This command executes the **sysadm** subcommand **installpkg**.

Step 2: Insert the Networking Support Utilities floppy diskette or tape and, when prompted, press <**CR**> as instructed. Once you have hit <**CR**>, your computer will display the full path names of the files as they are copied from the floppy diskette or tape to the hard disk. Be patient, this will take several minutes.

Step 3: When instructed, remove the floppy diskette or tape and store it with your other diskettes or tapes. You must then type **q** to return to the shell.

Step 4: As instructed in the *Procedure 6* section of the Administrator's Guide, rebuild the UNIX system kernel, install in / and reboot the machine as follows:

```
# shutdown -y -g0
press RESET button
```

Once you receive the `Console Login:` prompt your system will be ready, and the Networking Support Utilities installation is complete.

For more information on configuring STREAMS for your system, see Chapter 6, "Performance Management," in the *System Administrator's Guide*.

Software Notes

This section describes problems that may occur with the Networking Support Utilities and, in some cases, workarounds for these problems.

listen

If two or more simultaneous connection requests come in at the same time, the listener can only accept one; the others will be disconnected by the transport provider.

Client side networking code should be prepared to handle the case where their connection request fails because of a disconnect arriving on the stream. In this case the code should be designed to use some retry mechanism.

listen

The command line in the listener data base file (**/usr/net/nls/***net_spec*) is parsed using white space as the delimiter. Therefore, when building the argument list to pass to **exec**, quoted strings are not interpreted as one argument. This causes incorrect results when the server is the shell. For example:

/bin/sh -c "/bin/cat *filename* **>/dev/console"**

This problem can be worked around by using a shell script to perform the actual command:

/bin/sh -c /usr/net/servers/*shellscript*

Then *shellscript* would contain the following line:

/bin/cat *filename* **>/dev/console**

listen

The listener expects its protocol messages in one message, unless the **T_MORE** bit is set. If a client process sends the protocol message to the listener in chunks (for example, byte-by-byte), then the listener will not be able to assemble the message and it will disconnect the connection.

The client processes should always send the protocol message to the listener with one **t_snd** or one **write** call.

listen

When the listener process encounters an unrecoverable error, it exits silently. The error can be identified by tailing the end of the listener log file, **log**, which is found in **/usr/net/nls/***net_spec*. Because the log file is truncated each time the listener is started, it must be inspected before restarting the listener.

nlsadmin

The **nlsadmin** command allows the administrator to assign the same network address to the general listener service (using the **-l** option) and the login service (using the **-t** option) for the listener on a network. When that listener is subsequently started, it terminates with an error.

To avoid this problem, do not assign the same value to the listener service address and the login service address.

nlsadmin

If you change the address the listener is listening on with the **nlsadmin -l laddr -t taddr net_spec** command, the addresses must be entered without an embedded new-line between the start of the address and the terminating white space for that address. In other words, if you type:

nlsadmin -l *sftig.s*\
erve **-t** *sftig* **starlan**

then the address file for the listener will be in an invalid state and the listener will listen on names you are not expecting (namely, *sftig.s* and *erve*).

Do not break the address between two lines.

nlsadmin

When **nlsadmin** has trouble locating a *net_spec* or a data base file, two error messages are produced: *net_spec xxx invalid* or *net_spec xxx not found.* Both indicate that something is wrong [either the *net_spec* is invalid, or its corresponding file(s) are missing].

strclean

When **strclean** is used to remove *error.** files, there is no warning on failure. If the directory is not accessible to the user, the exit code indicates success even though it failed.

Do not rely on the exit codes of **strclean**. It may be good practice to verify that files have been removed properly after using this command.

STREAMS

A race condition exists in clone opens from different inodes. This problem exists when two or more disk inodes with the major of the clone device and equal minors are being opened at the same time. If the window is hit, then another open after the first open may bypass the clone device entirely, thus failing. For example, if */dev/node1* was major 63 and minor 57, and */dev/node2* was also major 63 and minor 57, and if they were two different inodes, then simultaneous opens of the two devices may result in failure of the second open.

If two or more separate files are needed on disk, they should be created as links to one disk inode, thereby closing the window. In the above example, */dev/node2* should be linked to */dev/node1* instead of being a separate inode.

sysdef

If there is not enough memory to allocate the buffers for streams, the operating system does not allocate any. However, **sysdef** reports that the number of streams buffers is the same as the amount requested in the master file, even though it is not.

In this case, the **strstat** option of **crash**(1M) can be used to accurately reflect the number of streams buffers allocated and free.

TIRDWR

The following problem occurs if **TIRDWR** is pushed on the stream: during the closing of a stream, the **TIRDWR** module may sometimes hang the process while waiting for a disconnect acknowledgement from the transport provider. This problem may show up when using **cu** across the network or when the server process and the client process exit at the same time.

t_snd

Invoking the **t_snd** routine with the **nbytes** argument set to -1 causes it to send an improperly structured message to the transport provider. More specifically, **t_snd** will call **putmsg**(2) with a data size of -1, which will cause **putmsg** to send down a Transport Interface message with only the control part and no data part. This is not a legal Transport Interface message.

Do not use a byte count of -1.

Uutry

When you are using the Basic Networking Utilities over a transport provider, if a remote system listens on an address different from that in the local *Systems* file, trying to **Uutry** to it results in the following error message:

Call failed: NO DEVICES AVAILABLE

This does not imply that there are no available devices on the local system. However, this does imply that **Uutry** has failed after opening a device and before achieving a connection.

This failure could be caused by a variety of problems including no devices available on the local system or the address in the *Systems* file being incorrect. To see the local devices that are in use, type **uustat -p**.

RFS and Basic Networking First Time Setup and Startup Instructions (80386 Only)

These instructions apply to using the Intel SBC 552 Ethernet Board and the iNA961 Firmware. There is also a section included about STARLAN on AT386. The purpose is to cover those areas that are either specific to Intel Architecture, or are not quite clear even to someone who has read the *System* Administration Guide.

Before we go any further, we should specify what an address is in this context. For Intel Architecture, using Intel i552 Ethernet board, the format of an address is:

\x0A490000<ethernet address>FE02<byte swapped TSAP>

where TSAP is the Transport Service Access point.

TSAP is 0001 for the general requests and 0002 for remote terminal requests, and is used by the listener to service remote requests. For example, a listener network address for a system with the Ethernet address of 00AA00004B43 and TSAP of 0001 is:

\x0A49000000AA00004B43FE020100

First Time Setup

Initially, the driver prints the Ethernet address such as 0–AA–00–4B43. To get the correct Ethernet address, you will have to pad each 0 (zero) to two 0s (zeros) such as 00–AA–0000–4B43. For other systems only the Ethernet part of the address would change. Notice that \ at the beginning of the address has to be escaped.

```
nlsadmin -i iso-tp4
# Create and initialize the files required by the listener

nlsadmin -l \\x0A49000000AA00004B43FE020100 iso-tp4
# Setup the address on which the listener will listen for general
requests (including RFS)

nlsadmin -t \\x0A49000000AA00004B43FE020200 iso-tp4
# Setup the address on which listener will listen for Remote Login
requests
```

```
nlsadmin -m -a1 -c "/usr/net/servers/tty/ttysrv tp4-" -y "Termi-
nal Server" iso-tp4
# Pass the service code for the Remote Login to the listener
```

In the above (nlsadm –m –a1 –c"), the 1 (one) in –a1 is the digit 1 (one) and should not be confused with the letter l.

Startup and Shutdown

Once the system is set up, starting Networking and RFS is as easy as changing *init*'s run level to run level 3. You can stop Networking and RFS by changing *init*'s run level to run level 2.

Network Startup Operation

Your system is configured to start up Networking and RFS on level 3. The following sequence of commands must be executed every time the network is brought up on a machine. These commands are executed automatically by **/etc/rc3.d/S10nls** script when *init*'s run level is changing to run level 3.

```
i552pump /dev/tp4-00 /etc/ina961.21
# Down loads the firmware for the i552 board, only when the
machine is brought up (booted)

nlsadmin -s iso-tp4
# Start listener for iso-tp4, every time the network is brought up
where iso-tp4 is the network special file cloneable driver, and tp4-
00 is the network special file.
```

ina961 binaries included are to be downloaded to Intel iSXM 552 and iSXM 554 boards. Five boot preconfigured boot modules are provided for the following configuration:

Boot Module	Hardware Type	Transport Layer	Network Layer	Virtual Circuits
ina961.21	iSXM552	yes	null	30
ina961.22	iSXM552A	yes	null	100
ina961.23	iSXM552A	yes	internet	100
ina961.24	iSBC554	yes	internet	100
ina961.25	iSBC554w/o iSBX586	no	internt	-

If you need to download any other version of *ina961* other than *ina961.21* , edit **/etc/rc3.d/S10nls** to download the correct version of firmware.

Stopping the Network

To stop the network you should execute:

```
nlsadmin -k iso-tp4
```

If the machine was not brought down, then the network could be restarted by just executing:

```
nlsadmin -s iso-tp4
```

and there is no need to run the *i552pump* command.

Automatic Startup of Network on Level 2

If you would like to have Networking in level 2, execute the following commands:

```
mv /etc/rc3.d/S10nls /etc/rc2.d
# move the startup script for networking to rc2.d
```

rm /etc/rc2.d/K67nls
remove the shutdown script for networking from rc2.d

vi /etc/rc2.d/S10nls
edit the rc script so that networking is started only if the last state was **SINGLE-USER.** The 'start' section of this file should look like the following:

```
'start')
        set 'who -r'
        if [ $9 = S ]
        then
        /bin/i552pump /dev/tp4-00 /etc/ina961.21
        /usr/bin/nlsadmin -s iso-tp4 2> /dev/console
        rm -f /usr/net/nls/log/* /usr/net/nls/tmp/*
        fi
    ;;
```

RFS and Basic Networking First Time Setup and Startup Instructions (80286 Only)

This information is to be provided at a later time.

Documentation

The following documents are provided with the Networking Support Utilities:

UNIX System V Intel 80386 Computer Version Networking Support Utilities Release 1.0 Release Notes (select code 307-642)

STREAMS Primer (select code 307-649)

The following Network Support Utilities documents are optionally orderable; see the section below "How to Order Documents":

STREAMS Programmer's Guide (select code 307-680)

Network Programmer's Guide (select code 307-641)

For further information on additional documentation for System V Release 3, see the *Product Overview* (select code 307-643) and the *Documentation Roadmap* (select code 305-555).

How to Order Documents

Additional copies of any document or optional documents can be ordered by calling the AT&T Customer Information Center:

1-800-432-6600 (toll free within the continental United States)

1-317-352-8557 (outside the continental United States)

or by writing to:

AT&T Customer Information Center
Customer Service Representative
P. O. Box 19901
Indianapolis, Indiana 46219

Preface

Software Description

Remote File Sharing (RFS) is a software package that allows computers running UNIX System V Release 3.0 to share resources selectively (files, directories, devices, and named pipes) across a network. Administrators for computers on an RFS network can choose directories on their systems they want to share and add them to a list of available resources on the network. From this list, they can choose resources from remote hosts that they would like to use on their computers.

Each host computer on a Remote File Sharing system can be grouped with others in a "domain" or operate as an independent domain. The domain can provide a central point for administering a group of hosts. Unlike other distributed file systems used with the UNIX operating system, Remote File Sharing is built into the operating system itself. This approach has several advantages:

Compatibility
: Once you mount a remote resource on your system it will look to your users as though it is part of the local system. You will be able to use most standard UNIX system features on the resource. Standard commands and system calls, as well as features like File and Record Locking, work the same on remote resources as they do locally. Applications should be able to work on remote resources without modification.

Security
: Standard UNIX system file security measures will be available to protect your resources. Special means for verifying host access to your resources and restricting remote users' permissions have been added for Remote File Sharing.

Flexibility
: Since you can mount a remote resource on any directory on your system, you have a lot of freedom to set up your computer's view of the world. You do not have to open up all your files to every host on the network. Likewise, you do not have to make all files on the network available to your computer's users.

Contents of the Release

The Remote File Sharing Utilities come on one floppy diskette. The files contained on that floppy are listed below.

/etc/conf/modules/du/du.o
/etc/conf/modules/du/space.o
/etc/conf/modules/du/stubs.o
/etc/conf/modules/dufst/dufst.o
/etc/conf/modules/dufst/config
/etc/conf/modules/sp/sp.o
/etc/conf/modules/sp/space.o
/etc/conf/modules/sp/config
/etc/rmount
/etc/rmountall
/etc/rumountall
/etc/init.d/adv
/etc/init.d/rfs
/etc/init.d/fumounts
/etc/init.d/rumounts
/usr/bin/adv
/usr/bin/rmntstat
/usr/bin/rfpasswd
/usr/bin/rfstart
/usr/bin/rfstop
/usr/bin/rfuadmin
/usr/bin/rfudaemon
/usr/bin/fumount
/usr/bin/fusage
/usr/bin/idload
/usr/bin/dname
/usr/bin/rfadmin
/usr/bin/nsquery
/usr/bin/unadv
/usr/nserve/auth.info
/usr/nserve/nserve
/usr/net/servers/rfs/rfsetup
/usr/options/rfs.name

Installation and Build Procedures

If you received this product as a separate package, and installation is required, this section describes the installation procedures for the Remote File Sharing Utilities.

Prerequisites

Before you can install Remote File Sharing Utilities, you must complete the following prerequisites.

Software

You must make sure the following software is installed before you install Remote File Sharing Utilities.

- Essential Utilities (System V Release 3.0)
- Directory and File Management Utilities
- Networking Support Utilities
- AT&T STARLAN NETWORK or some other AT&T Transport Interface-compatible network.

You must also have the following space available on your system:

- / (root file system) needs 863 blocks of free space.
- **/usr** needs 1110 blocks of free space.

NOTE

513 of the blocks needed in root are just temporary space that is used to reconfigure the system. After the package is installed, only about 350 blocks are used up.

Hardware

Remote File Sharing can be run on your computer. The following hardware is required:

- Minimum of 2 megabytes of memory. (If you are upgrading to 2 megabytes of memory, make sure you update your tunable parameters to match that amount of memory. See Chapter 6 of the *System Administrator's Guide* for more information.)
- Any communications hardware required by the network you are using.

Installation Procedure

Remote File Sharing is not currently packaged as a separate product; it is installed as part of the UNIX System V Release 3.0.

This procedure describes Remote File Sharing installation. The Remote File Sharing Utilities for your computer are distributed either on one floppy diskette or cartridge tape. Most of the utilities are object code files. However, some are shell scripts that you can modify to suit your operating environment. To begin this procedure, you must:

- Place your computer in system state—2 (multiuser) or 1 (single-user). You must **mount /usr** to run this procedure in single-user mode.
- Be at the computer to insert and remove diskettes or tape.
- Log in as **root**.

Run sysadm installpkg

Step 1: To install the Remote File Sharing Utilities, use the direct access method of the System Administration menu as follows:

```
# sysadm installpkg
```

This executes the sysadm subcommand **installpkg**.

Step 2: Insert the Remote File Sharing Utilities floppy diskette or tape and press **RETURN** as instructed. Once you have hit **RETURN**, your computer will display the full path names of the files as they are copied from the floppy diskette or tape to the hard disk. Be patient, this will take several minutes.

Step 3: When instructed, remove the floppy diskette or tape and store it with your other utilities diskettes. You must then type **q** to return to the shell.

Step 4: As instructed in the *Procedure 6* section of the Administrator's Guide, rebuild the UNIX system kernel, install in / and reboot the machine as follows:

```
# shutdown -y -g0
press RESET button
```

Once you receive the **Console Login:** prompt your system will be ready.

Remote File Sharing installation is complete. You must now configure your Remote File Sharing system as described in Procedures 10.1 through 10.4 in the *System Administrator's Guide*. Detailed information on Remote File Sharing is covered in Chapter 10 of the *System Administrator's Guide*.

Software Notes

This section describes problems that may occur with Remote File Sharing and, in some cases, workarounds to those problems.

acct

The accounting file passed to the **acct**(2) system call cannot be remote. This restriction applies to user software that uses the system call directly and to the software in the optional processing accounting package. RFS does not allow the **acct** system call; if passed a remote path name, **acct** will return an **errno** of **EINVAL**.

adv, unadv

Concurrent execution of the **adv** and **unadv** commands corrupts the advertise table. For example, executing the following two commands:

```
adv USR /usr &
unadv USR &
```

causes subsequent discrepancies in the resources printed by **adv** (which accesses the advertise table maintained on the local host) and **nsquery** (which accesses the advertise table maintained by the domain name server).

To avoid this, the **unadv** command should not be executed until all **adv**'s have completed.

cd

The following is a scenario for a multi-hop in an RFS environment: Machine A advertises / as *RES1* and machine B advertises / as *RES2*. Then A mounts *RES2* on **/mpd**, and B mounts *RES1* on **/mnt**. If B tries to **cd** to **/mnt/mpd**, the error message given is `bad directory`, instead of the appropriate multi-hop error message.

cu

The Basic Networking Utilities (BNU) package allows the use of remote devices for operations such as **cu**. However, if a user on machine alpha has a device **/dev/xxx** and remotely mounts a **/dev** directory from machine beta, which also has an **xxx** entry, any use of either device will cause the other line to appear locked. The problem here is that the lock files used by BNU (in **/usr/spool/locks**) are associated with individual devices through the **basename** of the device, and the BNU commands are unable to distinguish between devices that have the same name but are in different directories.

When remotely mounting **/dev** directories, check for entries duplicated in the local **/dev** directories. If there are duplicate names, follow these steps:

1. Make an alias name using the **ln** command. For example:

   ln /dev/culd0 /dev/*system-name***.culd0**

2. Change all references in the BNU files to the *new* alias name (for example, in **/usr/lib/uucp/Devices**).

This will enable BNU to work as well as any locally written commands that make use of the conventional names for networking devices.

cu

If RFS is in effect when machine A advertises devices (**/dev**) to other machines (for example, machine B and machine C), there is no way for **cu** to know when a device is being accessed remotely. Both machine B and machine C will mount machine A's **/dev** directory under a directory such as **/rdev**. If a user from machine B **cu**'s to a particular device, such as **/dev/xxx**, and a user from machine C tries to **cu** to that same device, both users will experience problems, for example, garbled information sent to the display screen. This problem will not occur if two users from the same machine try to access the same device, because the system locks the device for the first user. Unfortunately, the lock information resides only on the one machine; other machines in an RFS network have no way of knowing when a particular device is being accessed.

System administrators should be extremely cautious when advertising a directory like **/dev**.

df

If **df** is used without options, it lists each occurrence of a remote resource that is mounted on a system, and places an asterisk next to the word blocks for the second and each subsequent resource that was advertised under the same remote file system (for example, **/usr/mail** and **/usr/bin**). This signifies that the identical block counts for the resources reside under the same file system.

The problem is that if **df** is used with multiple remote resources passed as arguments, the asterisk never appears. In this example the asterisk does not appear:

```
$ df USRMAIL USRBIN
```

This problem can be misleading because the user may think that although the block and inode count for the two resources are identical, they are not part of the same file system, since the asterisk that indicates they are is not present.

fumount

The **-w** option to the **fumount** command allows a user to specify a grace period between warning clients that a resource is to be removed and actually removing the resource. The **atoi** subroutine [**strtol**(3C)] calculates the number of seconds. This routine looks for an initial numeric string and converts it to an integer. Any nonnumeric character in the argument terminates the argument. For example, the argument **-w 123abc** gives a grace period of 123 seconds. Missing arguments and arguments without an initial numeric string produce an error message.

fumount

The usage message from **fumount** says that the range of the wait time is 0 to 3600 seconds. The correct range is 1 to 3600 seconds.

fuser

The **fuser** command can fail to find processes that are using a resource, so an attempt to unmount the resource can fail because the resource is busy. For example, **fuser** does not find processes that are executing a text file in the target resource. Also, if a process tries to access a fifo, but blocks in **open**(2) waiting for a read or a write, **fuser** does not find the process. Finally, **fuser** may miss a process if that process gets a reference to the resource after **fuser** has begun its search.

In all these cases, the offending process can be killed explicitly with the **kill** command. When all processes using the resource are gone, the resource can be unmounted.

idload

When **idload** is run with the **-n** option, it shows how it will interpret the specified rules file. However, in showing the interpretation of a **default** statement, **idload** does not list the name of the user or group specified, even if the **default** statement identified the user or group by name rather than by numeric ID.

idload

In processing a rules file describing user ID or group ID mapping, **idload** will give a warning message if it encounters an ID in a **map** statement that previously appeared in an **exclude** statement. However, the warning that it gives is misleading, and says that the ID was previously mapped, rather than that it was previously excluded.

idload

When using the **map** instruction for the **idload** command, one can map a remote ID to a local numeric ID. However, if the local numeric ID is greater than 65535, the value that is actually used is the local ID modulo 65536. A valid mapping request on your computer would not include a local ID this large anyway, but such a request could be made due to a typing mistake or misunderstanding on the part of a system administrator.

idload

In the **idload** command, the **map** instruction may be given arguments of the form *X:Y*, meaning to map remote ID *X* into local ID *Y*. However, if the second ID is omitted (leaving **X:**), then remote ID *X* will be mapped into local ID 0 (root).

If you **exclude root** as recommended, the case described above will never happen. If root is excluded, *X:* would map *X* into ID number **MAXUID**+1 (60001 by default).

idload

If the **idload** command is given a **map** line longer than 256 characters, it fails with an unclear error message. The error message states that it found an invalid token *X*, where *X* is actually the last character processed on that line. The message should identify the problem as having been given too long a line.

If this problem occurs, the offending **map** line should be broken up into two or more lines.

labelit

On mounting a **ctc** device remotely under a local directory other than a subdirectory of **/dev**, **labelit** of the tape does not work on the remote machine because it expects the special file to begin with **/dev** and not with the path name on which the raw device was mounted.

logs

The following log files contain information relating to Remote File Sharing activities:

/usr/adm/rfuadmin.log
/usr/net/servers/rfs.log
/usr/net/servers/rfs/rfs.log
/usr/net/nls/*netspec***/log**

These files are for Remote File Sharing use only! Customers should not rely on the contents of these files because the information may change or the file may be deleted in future releases. Any tool written that takes advantage of the information contained in these files is not guaranteed to work in the future. (*netspec* above is replaced by the transport provider used by RFS. For the STARLAN NETWORK, this will be **starlan**.)

mount

When a mount fails because of a password mismatch, the error message can be confusing. The following error messages result from a remote mount failure due to mismatched passwords:

```
negotiate: An event requires attention
mount: negotiations failed
mount: possible cause: machine password incorrect
mount: could not connect to remote machine
```

mount

When a remote resource is disconnected by a **fumount** or a broken link, the default action in the (client) **rfuadmin** script is to try to remount the resource as it was mounted before. Therefore, if a resource that was originally read/write is readvertised read-only, the automatic mount will never succeed.

An administrator can always enter the **mount** directly using the latest advertised mode.

name server

When the primary and secondary name servers are under heavy load, the normal passing of name server information between these hosts may cause the machines to hang because the 1K Streams buffers have been depleted. There is one long term and one short term solution to the problem. For the long term, you can increase the number of 1K Streams buffers in **/usr/include/sys/kdef.h**. The parameter is NBLK1024. The short term solution is that you can stop Remote File Sharing on any secondary name server that is hung and then bring it back up again. That will clear the NBLK1024 buffers.

nsquery

The resource list printed by **nsquery** does not always reflect the current state of the domain. If a resource is advertised and the server goes down, a subsequent **nsquery** from a client may still list the resource as being available, even though it is not. An attempt to mount the resource will fail, because it is unable to contact the server.

nsquery

The **nsquery** command will return only the first 100 advertised resources even if there are more advertised.

process

The last process slot is traditionally reserved for a super-user process. However, an RFS server can take the last slot, making it impossible for any new process to start. If this server, or any other process, exits, as would normally happen, new processes can start.

programs

If a program creating remote directories or files loses its link to the remote machine, and the remote resource is unmounted, the program may begin to create *local* directories and files. For example, if you are using the **find** command piped to **cpio** to a remote machine and the link to the remote machine goes down and the resource is then unmounted, **cpio** may begin writing on the local machine—the target directory now looks just like an ordinary local directory.

ps

When a client mounts a remote executable file and runs it in the background, a **ps** command on the client machine will sometimes show a "null" process running or the mounted directory name as a running process. Using **ps -f** will give the correct information.

rfadmin

The command **rfadmin -r** *domain.machine* is used to remove the entry for *domain.machine* from the domain membership list. When this command is executed, the **rfmaster** file is checked to make sure that the machine being removed is not a backup (secondary) name server. However, there is no check to make sure that the machine being removed is not the primary name server. This can result in the entry for the primary machine being removed.

rfadmin

When executed with no options, the **rfadmin** command returns the name of the current domain name server. If this command is executed on a machine on which RFS is running, but no domain name server is available, the following message is printed:

`rfadmin: cannot obtain the name of the primary name server for domain` *dom*

If RFS is not running on the local machine, the same message is output, even if the domain name server is operating normally. This is misleading.

rfmaster

The acting domain name server is responsible for distributing important name service information to all other accessible (secondary) name servers that are serving the same domain, with no more than a 15-minute lag, so that if the acting name server should fail, another host could assume the name server role with a minimal loss of information. However, changes to the **rfmaster** file after **rfstart** has been run are not included in the information that is distributed in this way. Because the designation of hosts as primary and secondary name servers is made in the **rfmaster** file, this has the consequence of not allowing a change to the configuration of which hosts are the primary and secondary name servers for a domain without stopping and re-starting RFS on the affected hosts. [For example, adding a new secondary name server to the **rfmaster** file will not take effect until RFS is taken down on all of the existing (primary and secondary) name servers, as well as the newly designated secondary and then re-started.]

This limitation should not be confused with the temporary transfer of name server responsibility to another one of the hosts already listed in the **rfmaster** file as a primary or secondary name server; this temporary transfer is performed with the **rfadmin -p** command.

rfpasswd

The **rfpasswd** command is used to change the host password used for RFS and is intended to parallel the **passwd** command in the way it prompts for old and new passwords. However, if a host has no password (for example, it has a null password), the **rfpasswd** command will still prompt for the old password before asking for the new one, although it should ask only for the new one. Furthermore, the prompt for the old password is `Password`, although it should be `Old password`.

rfs

If the administrator of a server machine explicitly kills all of the server processes, all of the client processes that have requests on a server will hang, forever waiting for a response from the server.

rfstart

RFS is initiated on a machine by executing the **rfstart** command. Under some circumstances, **rfstart** will prompt for a password. In this case, RFS has actually been started on the local machine before **rfstart** completes, and RFS commands executed on another terminal can succeed while **rfstart** is still waiting for the password to be entered. However, if any resources are advertised from the local host before **rfstart** completes, those advertises will be erased from the local advertise table when the **rfstart** does complete.

This problem can be avoided by not issuing additional RFS commands until the **rfstart** completes and exits to the shell.

rfstart

The **rfstart** command can take up to a minute or more to return when executed from the shell, depending on response from the network. When possible, **rfstart** should be run automatically by using **init 3**.

rfudaemon

User-level recovery of resources that are disconnected gracefully (the remote system shuts down) may fail if the number of lost resources exceeds half of the value of the tunable parameter **MAXGDP** in **/usr/include/sys/kdef.h**. By default, **MAXGDP** is 24. The failure is accompanied by one or more of the following messages:

```
rfs user-daemon queue overflow:
    make sure rfudaemon is running
```

rmount

The **/etc/rmount** script loops forever if the system administrator manually mounts the remote resource during the **rmount** sleep interval. If this occurs, do a **ps -ef** to get the process ID of **rmount**, and then kill the process manually by issuing **kill -9** *pid#*.

stat

Some commands, such as **ls -l**, can be used to identify which users have access to a file. These commands obtain data on a file via the **stat** system call. When referencing a remote file, the ownership information returned by **stat** is converted to local UIDs/GIDs by computing the inverse of the UID/GID mapping on the server machine.

Under some UID/GID mapping specifications, it will erroneously appear that local users have access to a remote file when, in fact, the remote file is inaccessible to all local users. For example:

UID Mapping:

```
global
default transparent
exclude 0
```

Command:

```
$ ls -l /remote/etc/passwd
-rw-r--r--  1 root  sys   32449 Jan 22 19:40 /remote/etc/passwd
```

Despite what is reported by the **ls** command in this example, the local user **root** (UID=0) cannot write the remote file **/remote/etc/passwd**.

sticky bit

If a program that has the sticky bit set is shared through RFS and is executed by a user on a client machine, then it becomes impossible to remove the program (for example, with the **rm** command). Attempts to remove the file will generate an error message of "text busy," even though no one on any machine is currently running the program. Removing the sticky bit and rerunning the program has no effect. A program with the sticky bit set that has never been run across RFS can be removed without complaints, even if it has been run locally.

To remove such a "stuck" sticky-bit program, it is necessary to unmount the remote resource. This can be done either from the server, with the **fumount** command, or from the client, with the **umount** command.

streams

The three system calls related to STREAMS, **getmsg**, **putmsg**, and **poll**, will not operate with a file descriptor associated with a remote file. If this is attempted, the system call will fail with **errno** equal to **EINVAL**.

swap

Swap devices cannot be remote. This includes the swap device configured initially, and any swap devices added using the **swap**(1M) command.

umount

If a client loses its link to a server, any attempt to **umount** one of that server's file systems from the client tree will fail until recovery runs. Recovery from a link failure is handled by **rfuadmin**(1M) and **rfudaemon**(1M).

Recovery runs automatically when the link breaks, but not until someone tries to access the link, or until at most 11 minutes have passed. (The 11-minute time interval applies if you are using STARLAN NETWORK. The time may be different for other transport providers.)

If the **umount** fails because the link is gone, the **unmount** will start recovery. After recovery runs, a second **unmount** will succeed.

unadv

If **unadv** is invoked for a resource that was not advertised from this host, and this host is not the acting domain name server, then the command fails (as it should), but with a vague error message:

```
unadv:  Permission for operation denied
```

RFS First Time Setup and Startup Instructions (80386 Only)

These instructions apply to using the Intel SBC 552 Ethernet Board and the iNA961 Firmware. There is also a section included about STARLAN on AT386. The purpose is to cover those areas that are either specific to Intel Architecture, or are not quite clear even to someone who has read the *System Administration Guide*.

Before we go any further, we should specify what an address is in this context. For Intel Architecture, using Intel i552 Ethernet board, the format of an address is:

\x0A490000<ethernet address>FE02<byte swapped TSAP>

where TSAP is the Transport Service Access point.

TSAP is 0001 for the general requests and 0002 for remote terminal requests, and is used by the listener to service remote requests. For example, a listener network address for a system with the Ethernet address of 00AA00004B43 and TSAP of 0001 is:

\x0A49000000AA00004B43FE020100

First Time Setup

The machine to be set up is called *scrooge* and its Ethernet address is 00AA00004B43. You have to first set up the **/usr/nserve/rfmaster** file on the machine that is going to be the primary name server site. **Rfmaster** contains a reference to the domain name server in the following format. (Note: fields in brackets <> are variable fields; the others are constant):

<domain>	p	<domain.ns>
<domain.ns>	a	<address>

<domain> = the name of this machine's domain.

p = keyword for record that defines a domain name server.

<domain.ns> = name of domain name server for <domain>

a = keyword for record that defines address of a machine.

<address> = listen network address of machine <domain.ns>, for Intel i552 Ethernet board, the format of the address is :

\x0A490000<ethernet address>FE02<byte swapped TSAP>.

TSAP is the Transport Service Access point, which is 0001.

So for *scrooge* to be the primary domain name server of the domain *ism*, the following lines have to be in the rfmaster file on the *scrooge*:

```
ism           P     ism.scrooge
ism.scrooge   A     \x0A49000000AA00004B43FE020100
```

Note: Do not intermix spaces and tabs in the rfmaster file. The domain name must be less than 14 characters, or problems may occur.

dom.master is another file in the same format that contains information about other domains. This file is optional, and if used at all, will only be found on the primary name server.

The following is the sequence of the commands that has to be executed on all the machines the first time, before you can bring RFS up on them. The normal sequence, starting from the beginning for the machine *scrooge* in the domain *ism*, is given as an example. For other machines, and/or domains, only the Ethernet part of the address and the machine and/or the domain names would change.

```
dname -D ism
# Set the machine's domain name

nlsadmin -a105 -c/usr/net/servers/rfs/rfsetup -y "RFS Server"
iso-tp4
# Pass the service code for the RFS to the listener

dname -N iso-tp4
# /usr/nserve/netspec should contain the name of the special dev-
ice for the network.

rfadmin -a ism.scrooge
# set up the passwd file

echo "<passwd>\c" > /usr/nserve/loc.passwd
# /usr/nserve/loc.passwd has to exist for RFS automatic startup.
<passwd> is the same password supplied to rfadmin.
```

Startup and Shutdown

Once the system is set up, starting Networking and RFS is as easy as changing *init*'s run level to run level 3. You can stop Networking and RFS by changing *init*'s run level to run level 2.

RFS First Time Setup and Startup Instructions (80286 Only)

This information will be provided at a later date.

Documentation

The two documents you will need to set up and run Remote File Sharing are:

- *UNIX System V Intel 80286/80386 Computer Version Release 3.0 System Administrator's Guide* (select code 307-681).
- *UNIX System V Intel 80286/80386 Computer Version Release 3.0 System Administrator's Reference Manual* (select code 307-682).

For specific network information, refer to the *Documentation Roadmap* (select code 305-555).

For further information on additional documentation for System V Release 3, see the *Product Overview* (select code 307-643) and the *Documentation Roadmap* (select code 305-555).

How to Order Documents

Additional copies of any document or optional documents can be ordered by calling the AT&T Customer Information Center:

1-800-432-6600 (toll free within the continental United States)
1-317-352-8557 (outside the continental United States)

or by writing to:

AT&T Customer Information Center
Customer Service Representative
P. O. Box 19901
Indianapolis, Indiana 46219

Other books in the Prentice Hall C and UNIX® Systems Library

- The C Programmer's Handbook **Bell Labs/M. I. Bolsky**
- The UNIX System User's Handbook **Bell Labs/M. I. Bolsky**
- The Vi User's Handbook **Bell Labs/M. I. Bolsky**
- UNIX System Software Readings **AT&T UNIX PACIFIC**
- UNIX System Readings and Applications, Volume I **Bell Labs**
- UNIX System Readings and Applications, Volume II **Bell Labs**
- UNIX System V/386 Utilities Release Notes **AT&T**
- UNIX System V/386 Streams Primer **AT&T**
- UNIX System V/386 User's Guide, Second Edition **AT&T**
- UNIX System V/386 User's Reference Manual **AT&T**
- UNIX System V/386 Programmer's Reference Manual **AT&T**
- UNIX System V/386 Streams Programmer's Guide **AT&T**
- UNIX System V/386 Network Programmer's Guide **AT&T**
- UNIX System V/386 Programmer's Guide **AT&T**
- UNIX System V/386 System Administrator's Guide **AT&T**
- UNIX System V/386 System Administrator's Reference Manual **AT&T**

PRENTICE HALL, Englewood Cliffs, N.J. 07632

ISBN 0-13-936121-9