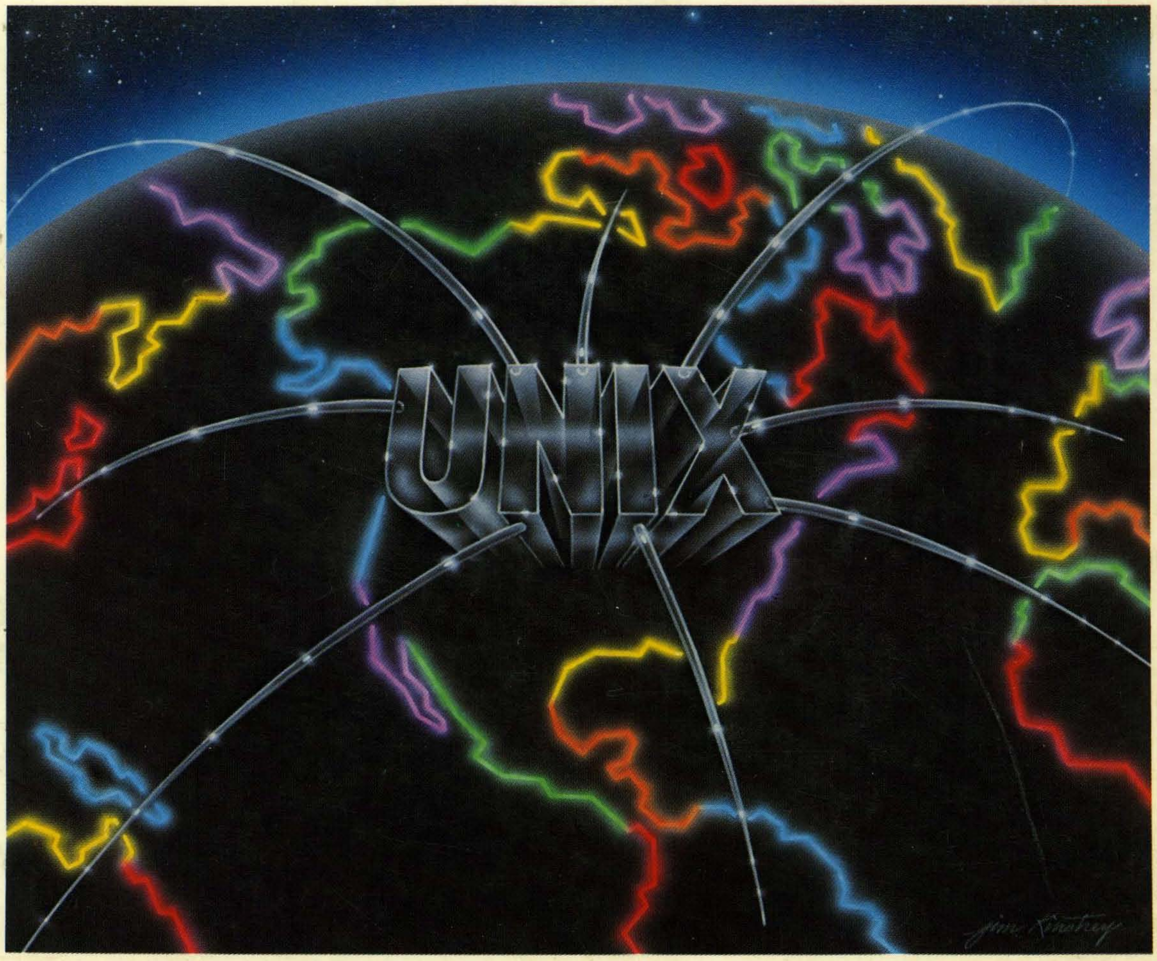
AT&T

UNIX® System V

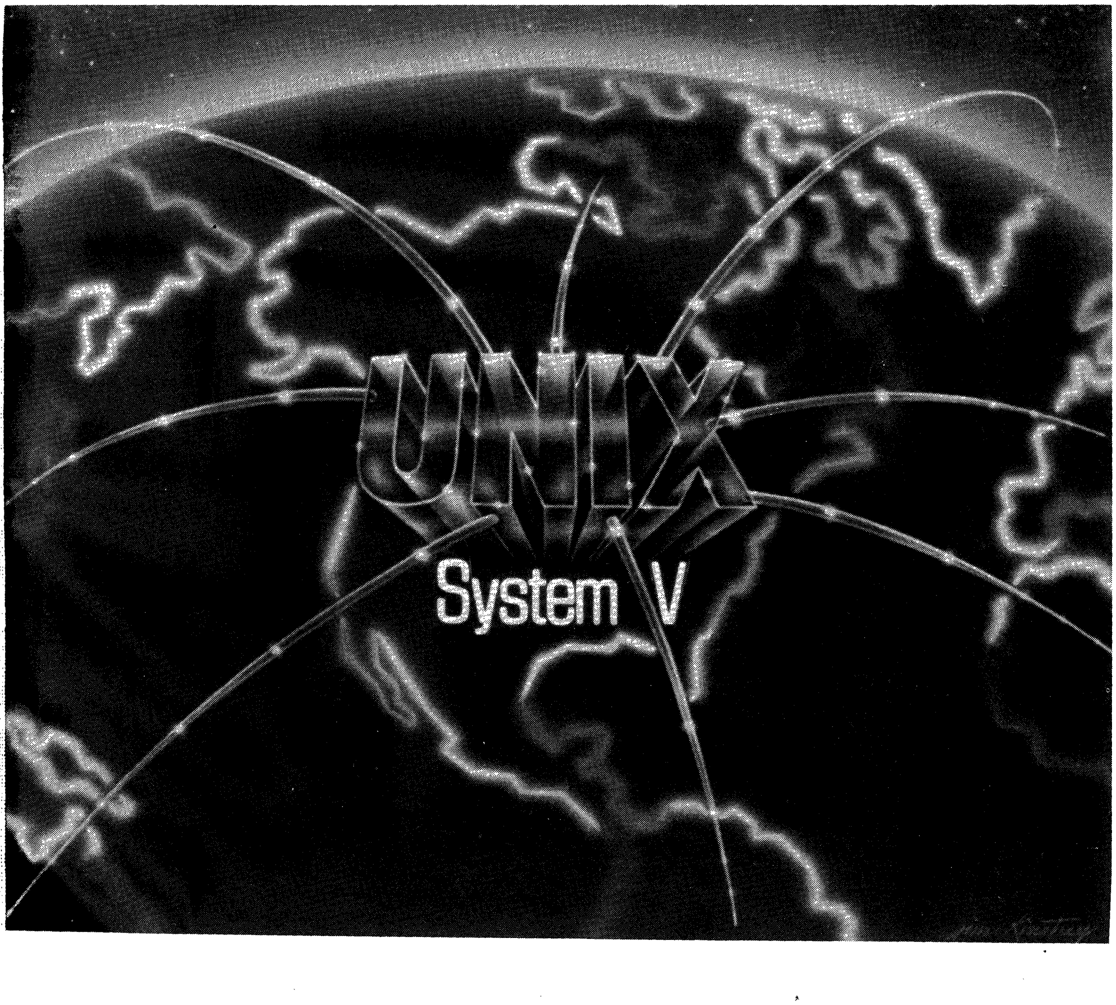
UTILITIES RELEASE NOTES



AT&T

ERRATUM
(CORRECTS COVER GRAPHIC)

UNIX® System V



UNIX® System V

Utilities Release Notes

AT&T

UNIX® System V

Utilities Release Notes

AT&T

Prentice-Hall, Inc., Englewod Cliffs, NJ 07632

Library of Congress Catalog Card Number: 87-60151

Editorial/production supervision: *Karen S. Fortgang*
Cover illustration: *Jim Kinstry*
Manufacturing buyer: *S. Gordon Osbourne*

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-940552-6 025

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*
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*
Prentice-Hall of Southeast Asia Pte. Ltd., *Singapore*

Ci Col.
QA
76
.76
.O63

Table of Contents

UNIX® System V

Utilities Release Notes

UNIX SYSTEM V RELEASE 3.0 RELEASE NOTES

Preface

These *Release Notes* contain important information about UNIX System V Release 3.0 on AT&T 3B2 Computers. First, they briefly describe the new features of this release. Next, they tell you how to upgrade from earlier releases of UNIX System V and how to change from a single-disk to a dual-disk system. Software notes, additional information about upgrading and 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 AT&T 3B2 Computers, see the *Product Overview*. For a complete description of the procedures used in the administration of an AT&T 3B2 Computer running UNIX System V Release 3.0, see the *System Administrator's Guide* (AT&T).

Additional Products

Two products are offered separately from UNIX System V Release 3.0: 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* (AT&T) and the *Networking Support Utilities Release Notes* (AT&T).

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 *UNIX System V User's Reference Manual* (P-H), unless otherwise noted. Manual pages from sections (3) and (4) appear in the *Programmer's Reference Manual* (P-H). Manual pages from section (1M) appear in the *System Administrator's Reference Manual* (AT&T).

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 login 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 comprises many packages. A product is something that you purchase independently of UNIX System V Release 3.0, for example, Remote File Sharing.

Upgrading to UNIX System V Release 3.0

In contrast to a Full Restore, which erases everything on the hard disk, an upgrade changes specific data (for example, kernel parameters) while preserving user files. To upgrade a system running UNIX System V Release 2.0 version 2, Release 2.0.4, or Release 2.1 to this release, please follow "Upgrade Procedure 1: Release Upgrade" below. To change your 3B2 Computer from a single-disk to a dual-disk system, first upgrade your system to Release 3.0, and then follow "Upgrade Procedure 2: Dual-Disk Upgrade." To do a Partial or Full Restore of your system, refer to the *System Administrator's Guide* (AT&T).

The System Administration menus help you set up the system. The **sysadm** command executes these menus during the upgrade of the utilities packages that come with UNIX System V Release 3.0. Chapter 4, "System Administration Menus," of the *Owner/Operator Manual* (AT&T) explains the System Administration menus and the **sysadm** command in detail; to set up the system for the first time, please refer to the this manual.

Features of UNIX System V Release 3.0

This release maintains the compatibility of source code across the 3B computer family (with the exceptions noted below), and object code between 3B2 and 3B5/3B15 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. However, you must have Release 2.3 of the 3BNET Local Area Network to use it with UNIX System V Release 3.0.

Dot-Mapped Display (DMD) software Release 1.2 and Release 2.0 are supported with UNIX System V Release 3.0. You must install DMD software before installing the AT&T Windowing Utilities Package.

If you have DEMON (DEbug MONitor) firmware, you may need new PROMs (Programmable Read-Only Memory). See the section "Additional Upgrade Information" for details.

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

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

Remote File Sharing

NOTE

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

NOTE

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 Organization for Standardization (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, see these release notes and the *System Administrator's Guide.*

Enhanced Basic Networking Commands

NOTE 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.

If you do buy the Networking Support Utilities, you 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 3-10 of the UNIX system command syntax standard. (The standard is described on the **intro**(1) manual page.)

NOTE 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.

Encryption Mechanisms Repackaged

This release introduces several changes in the implementation of the encryption mechanisms. In particular, the text editors that are part of the "Editing Utilities" package are no longer duplicated in the "Security Administration Utilities" package.

NOTE The Security Administration Utilities package has restricted distribution and is provided only with 3B2 Computers sold within the United States.

Upgrade Procedures

There are two upgrade procedures available for UNIX System V Release 3.0:

1. The first, a Release Upgrade procedure, allows you to upgrade an earlier UNIX System V release or to restore a system if its Shared Libraries become damaged without loss of any user files.
2. The second, a Dual-Disk Upgrade procedure, allows you, after Release Upgrade, to convert your 3B2 Computer from a single-disk to a dual-disk system without loss of any user files.

Two other important procedures are described in the *System Administrator's Guide*: one lets you partially restore the system to bring it to a usable state or to remove a forgotten **root** password, and the other lets you fully restore the system if there is a new system or disk, or if you need to increase the size of the swap, **/** or **/usr** partition by repartitioning the disk. (See "Partial System Restore" and "Full System Restore" in Procedure 3.9: Reload the Operating System.)

Six core system diskettes labeled "Essential Utilities Disk – 1" through "Essential Utilities Disk – 6" contain the basic operating software for UNIX System V Release 3.0. (The terms "core system" diskettes and "Essential Utilities" diskettes are synonymous.) Each of the upgrade procedures transfers files from diskette to the hard disk. The file transfer is automated by menu-driven programs. Adhere to the following instructions carefully to guarantee a smooth upgrade.

Upgrade Procedure 1: Release Upgrade

Purpose	To upgrade a system running UNIX System V Release 2.0 version 2, UNIX System V Release 2.0.4, or UNIX System V Release 2.1 to Release 3.0, or to restore a system if its Shared Libraries become damaged.
Machine	3B2 Computers.
Current Release	Any version of System V Release 2.
Media	The six 3B2 Essential Utilities diskettes.
Time	1-2 hours.

The Release Upgrade procedure allows you to upgrade your current release to Release 3.0. Here are the results of the procedure:

- The new core system is installed from the six diskettes to the hard disk.
- Certain system and user configuration files are moved temporarily to **/usr/old**. When the upgrade is finished, most of these files are moved back to where they were. See Step 7 below for a list of files that you must check after the upgrade.
- The **boot** programs are updated.
- The labels for **root** (/) and **/usr** are changed to reflect Release 3.0.

Before You Upgrade

CAUTION Do not attempt this procedure when sudden loss of power is likely (for example, during electrical storms). Loss of power during certain parts of the upgrade may cause loss of user data.

CAUTION It is highly recommended that you backup the hard disk(s) on floppy or cartridge tape before the Release Upgrade. (See Procedure 5.4, "File System Backup and Restore," in the *System Administrator's Guide.*)

You should backup individual login directories and save the following files (if they exist on your system) on floppies or cartridge tape before any core package installation:

/dgn/edt_data
/etc/fstab
/etc/gettydefs
/etc/group
/etc/inittab
/etc/master.d/kernel
/etc/master.d/msg
/etc/master.d/sem
/etc/master.d/shm
/etc/passwd
/etc/profile
/etc/system
/usr/lib/uucp/Systems
/usr/lib/uucp/Devices
/usr/lib/uucp/Poll
/usr/lib/uucp/Permissions

Once the automated Release Upgrade procedure from Essential Utilities Disk 1 begins, do not interrupt the process.

- If you are installing files from the first core floppy and the upgrade is interrupted for any reason, restart the upgrade from the beginning.
- If you are installing files from the second, third, fourth, or fifth core floppy and the upgrade is interrupted for any reason, you should be able to boot from the hard disk and continue where you left off.
- When you are installing files from the sixth core floppy, if the upgrade is interrupted for any reason after you have received the following message

```
You may now remove the last 3B2 Core System floppy
```

but before you receive the following message

```
Installation is now complete
```

you must restart from the beginning.

Any time that the upgrade fails, do a Full System Restore (see Procedure 3.9: Reload the Operating System in the *System Administrator's Guide*). With a Full System Restore, all user files are lost unless you have backed them up.

Upgrading Your Current Release

Here are the steps to upgrade your current release to Release 3.0:

Step 1 Login as **root**. Check the available space in **/** and **/usr**, for example, with **sysadm diskuse** or with **df −t**(1M). (Section (1M) is in the *System Administrator's Reference Manual.*) You must have at least 800 blocks free in **/** and 600 blocks free in **/usr** to do a Release Upgrade. You need more space than this to reinstall all the other packages and products available.

Enter **sysadm firmware** to the shell prompt to bring the system down to firmware mode, and then go to firmware in express mode (answering y to the appropriate prompt, as below). (This example shows upgrading from UNIX System V Release 2.1.)

```
Console Login: root <CR>
Password: <password> <CR>
UNIX System V Release 2.1.0 AT&T 3B2 Version 2
unix
Copyright (c) 1984 AT&T
All Rights Reserved
# df -t<CR>
.
.
# sysadm firmware <CR>

Running subcommand 'firmware' from menu 'machinemgmt'.
MACHINE MANAGEMENT

Once started, this procedure CANNOT BE STOPPED.
Do you want to go to firmware "express"? [y, n, q, ?] y <CR>

Shutdown started.   Thurs Jul 3 12:49:31 EDT 1986

Broadcast Message from root (console) Thurs Jul 3 12:49:34 ...
THE SYSTEM IS BEING SHUT DOWN NOW ! ! !
Log off now or risk your files being damaged.

INIT:  New run level:  5
The system is coming down.  Please wait.
System services are now being stopped.

The system is down.

SELF-CHECK

FIRMWARE MODE
```

Step 2   Insert diskette 1 of the Essential Utilities Disks into the diskette drive.

CAUTION Make sure that diskette 1 is not write protected.

Step 3 Enter the firmware password (**mcp**) or new firmware password (if you have changed it).

password

Step 4 Enter **unix** in response to the following prompt:

```
Enter the name of program to execute [ ] unix<CR>
```

Step 5 Enter **0** in response to the following prompt:

```
Enter Load Device Option Number [1, (HD30)] 0<CR>
```

(The default device displayed may not be HD30, but you may ignore this discrepancy safely.)

Step 6 After copyright information is displayed, you are asked to enter one of the following options:

1. Full Restore
2. Partial Restore
3. Dual-Disk Upgrade
4. Release Upgrade

Since you are performing a release upgrade to Release 3.0, enter option 4 in response to the following prompt:

```
Selection? [ 1 2 3 4 quit help ] 4<CR>
```

Step 7 From this point, the installation program is interactive; it prompts you through the release upgrade procedure. Install the diskettes in sequence. Leave diskettes 2 through 6 write protected. You can enter **help** when you want additional information, or you can enter **quit** to stop the upgrade procedure. If you enter **quit** and later reboot the system, the system will prompt you to pick up where you left off in the upgrade.

The Release Upgrade procedure puts the following configuration files in **/usr/old**, and then does one of three things with them:

1. upgrades them and then moves them back
2. keeps them there, replacing them with new versions
3. puts new versions of them in different locations

Here are files that may be upgraded by this procedure, depending on what release you upgrade from.

/etc/gettydefs
/etc/inittab
/etc/profile

Here are files that are put into **/usr/old** and kept there; the Release Upgrade puts new versions of the files where they had been:

/bin/ed
/bin/red
/etc/system
/etc/master.d/hdelog
/etc/master.d/iuart
/etc/master.d/idisk
/etc/master.d/kernel
/etc/master.d/mem
/etc/master.d/stubs
/etc/master.d/gentty
/etc/master.d/s5
/etc/master.d/ports

Here are files, new for UNIX System V Release 3.0, that are put into **/usr/old** and kept there if you do a Release Upgrade from a system already running Release 3.0 (for example, to restore a system if its Shared Libraries become damaged).

/etc/master.d/disp
/etc/init.d/ANNOUNCE
/etc/init.d/MOUNTFSYS
/etc/init.d/README

/etc/init.d/RMTMPFILES
/etc/init.d/autoconfig
/etc/init.d/cron
/etc/init.d/disks
/etc/init.d/firstcheck
/etc/init.d/sysetup
/etc/init.d/uucp
/etc/rc0.d/K00ANNOUNCE
/etc/rc0.d/K70cron
/etc/rc2.d/S00firstcheck
/etc/rc2.d/S01MOUNTFSYS
/etc/rc2.d/S05RMTMPFILES
/etc/rc2.d/S10disks
/etc/rc2.d/S15autoconfig
/etc/rc2.d/S20sysetup
/etc/rc2.d/S70uucp
/etc/rc2.d/S75cron

Here are files that are put into **/usr/old** and are not replaced; if these files already exist on your machine (which depends on the release that you are running before you upgrade), the system puts new versions of the files in a different location:

/etc/shutdown.d/ANNOUNCE
/etc/rc.d/0_firstcheck
/etc/rc.d/ANNOUNCE
/etc/rc.d/MOUNTFILESYS
/etc/rc.d/README
/etc/rc.d/RMTMPFILES
/etc/rc.d/acct
/etc/rc.d/autoconfig
/etc/rc.d/cron
/etc/rc.d/disks
/etc/rc.d/ports
/etc/rc.d/sysetup
/etc/rc.d/uucp

NOTE

Directories created in **/usr/old** by the Release Upgrade procedure are given mode 777. Also, **/usr/old/README** is given mode 666. This allows any user to remove the files left in **/usr/old** by the upgrade procedure. Some file modes and owners differ depending on whether you do a "Release Upgrade" or "Full Restore." These differences arise because the old files are kept during a "Release Upgrade." The following files are affected:

/dgn/edt_data
/etc/fstab
/etc/gettydefs

Step 8 After you have installed all files from the core floppies, the `Console Login:` prompt will reappear. Log in as **root**. Run **shutdown**(1M), and then mount all file systems with **mountall**(1M). (Section (1M) is in the *System Administrator's Reference Manual.*)

Remove old utility packages with **sysadm removepkg**. Use the floppies that you used to install the package originally.

Then, install utilities packages with **sysadm installpkg**, which starts another interactive procedure, and tells you if you load utilities out of order. You must have the floppies available for the packages that you want to replace.

CAUTION

If you do not remove packages before you re-install them, you may not have enough space for the new versions, or the installation may fail, or both.

Install the System Administration Utilities first, then the Directory and File Management Utilities. Install the remaining utilities packages in the order suggested by Figure 1 below. For details about this installation procedure, see sections describing Software Utilities Packages in the *Owner/Operator Manual.*

When you install a package that contains a driver, a message containing the following text is displayed at the end of the installation procedure:

```
Execute "shutdown -i6 -g0 -y"
```

During a Release Upgrade, Full Restore, or Partial Restore, you do not have to execute **shutdown −i6 −g0 −y** after installing each package. You may wait until all packages are installed to run **shutdown**.

NOTE

You must reinstall the following packages if you have them:

System Administration Utilities
Directory and File Management Utilities
User Environment Utilities
Inter-Process Communication Utilities
Editing Utilities
Performance Measurement Utilities
Security Administration Utilities (domestic customers only)
AT&T Windowing Utilities
Cartridge Tape Utilities
  (if your system is equipped with a Cartridge Tape Controller)
Networking Support Utilities (if you have it)
Remote File Sharing Utilities (if you have it)

In addition, you must reinstall any other packages that install a device driver. Dependencies between System V Release 3.0 utilities are described in Figure 1 below.

Utilities	Software Dependencies	Drivers Installed	Free Blocks Needed in / (**root**)	Free Blocks Needed in **/usr**
System Administration	None	No	700	10
Directory and File Management	None	No	190	920
User Environment	Directory and File Management	Yes	190	670
Inter-Process Communication	None	Yes	210	30
Terminal Filters	None	No	20	170
Terminal Information	None	No	170	1200*
Graphics	User Environment Terminal Filters	No	20	4000
Basic Networking	User Environment	No	20	1480
Editing	None	No	20	480
Help	User Environment Terminal Information	No	30	1100
Line Printer Spooling	User Environment	No	20	740
Performance Measurement	None	Yes	70	520
Security Administration	Terminal Information	No	90	70
Spell	Directory and File Management	No	20	470
AT&T Windowing	None	Yes	110	400

Figure 1: System V Release 3.0 Utilities Installation Chart

* The number of free blocks in **/usr** needed when files provided on the first floppy of the package are installed. If you install all files on the second floppy, an additional 1390 blocks are needed in **/usr**.

Step 9	After you finish installing utility packages, compare (for example, with the **diff** command) configuration files in **/etc/master.d** and **/etc/system** (those not saved and replaced) with their copies that still reside in **/usr/old** (See Step 7). Some differences between the old and new files may be because of your tuning rather than the upgrade. There also can be differences because of new and deleted tunables with Release 3.0. Edit the new files rather than copying them from **/usr/old**.
Step 10	Reboot the system by typing **shutdown –i6 –g0 –y**.

Upgrade Procedure 2: Dual-Disk Upgrade

Purpose	To convert the system from a single-disk to a dual-disk system.
Machine	3B2 Computers.
Current Release	Release 3.0
Media	Only the first diskette of the six 3B2 Essential Utilities diskette.
Time	40 to 90 minutes.

The Dual-Disk Upgrade procedure allows you to convert the system on your 3B2 Computer from a single-disk to a dual-disk system after you have done a Release Upgrade. This procedure does not upgrade your system; use the Release Upgrade procedure to bring your system to UNIX System V Release 3.0 before doing a Dual-Disk Upgrade.

NOTE

If you are using an AT&T 3B2/300 Computer or an AT&T 3B2/310 Computer, the Dual-Disk Upgrade procedure assumes that you have installed an XM (eXpansion Module, allowing for expanded I/O capability) with an additional hard disk drive. This is the only way to have a dual-disk system with these computer models. Previous 3B2 Computer UNIX System V releases required the installation of the XM Administration Utilities to use an XM. In System V Release 3.0, the XM Administration software is part of the Essential Utilities package. **sysadm** prevents installation of the old XM Utilities.

Here is a summary of the results of the Dual-Disk Upgrade procedure:

- The **/usr** file system is optionally moved to the second disk, allowing the free space on the first disk to be reallocated to partitions 8 through f. By default, the Dual-Disk Upgrade puts **/usr** on the second disk, and a new file system, **/usr2** is put on the first disk.
- The remaining space on the second disk is reallocated to partitions 8 through f.

Before You Do a Dual-Disk Upgrade

CAUTION Do not attempt this procedure when sudden loss of power is likely (for example, during electrical storms). Loss of power during certain parts of the upgrade causes loss of user data.

Once the automated Dual-Disk Upgrade procedure from Essential Utilities Diskette 1 begins, do not interrupt the process. Any interruptions, such as disk error, the RESET button pushed, or power failure, may make the system unable to boot from the hard disk. If this happens, you have to start over and do a Full System Restore (see Procedure 3.9: Reload the Operating System in the *System Administrator's Guide*). With a Full Restore, all user files are lost.

CAUTION Current backups of the hard disk are highly recommended before the Dual-Disk Upgrade so that recovery is possible.

When partitioning a disk during the Dual-Disk Upgrade, the minimum block count allowed for a partition depends on the cylinder size of the disk. For example, the minimum block count for a 32 megabyte disk is 90, the minimum block count for a 72 megabyte disk is 162. Partition sizes are always rounded to cylinder boundaries. The Dual-Disk Upgrade silently enforces these limits.

CAUTION

If during a dual-disk upgrade you quit the repartition after you have moved **/usr** to the second disk, but before you have put the remaining space of either the first or second disk into new partitions, the remaining space becomes inaccessible. In these circumstances, **sysadm partition** does not allow you to partition the disk; you must use **fmthard –d** to put the remaining space into a partition (see **fmthard**(1M)).

Converting Your System to a Dual-Disk System

To convert the system to a dual-disk system, follow these steps:

Step 1 Login as **root**. Enter **sysadm firmware** to the shell prompt to bring the system down to the firmware mode, and then go to firmware express mode (answering **y** to the appropriate prompt).

```
Console Login:  root <CR>
Password:  <password> <CR>
UNIX System V Release 3.0 AT&T 3B2
unix
Copyright (c) 1984 AT&T
All Rights Reserved
# sysadm firmware <CR>

Running subcommand 'firmware' from menu 'machinemgmt'.
MACHINE MANAGEMENT

Once started, this procedure CANNOT BE STOPPED.
Do you want to go to firmware "express"? [y, n, ?, q] y<CR>

Shutdown started.   Wed Jan 29 01:22:31 EDT 1986

Broadcast Message from root (console) Wed Jan 29 01:22:32 ...
THE SYSTEM IS BEING SHUT DOWN NOW ! ! !
Log off now or risk your files being damaged.


INIT:  New run level:  5
The system is coming down.  Please wait.
System services are now being stopped.

The system is down.

SELF-CHECK

FIRMWARE MODE
```

Step 2 Insert the copy of diskette 1 of the Essential Utilities Disks into the diskette drive.

CAUTION Make sure that diskette 1 is not write protected.

Step 3 Enter the firmware password (**mcp**) or new firmware password (if you have changed it).

password

Step 4 Enter **unix** in response to the following prompt:

```
Enter the name of program to execute [ ]: unix
```

Step 5 Enter **0** in response to the following prompt:

```
Enter Load Device Option Number [1, (HD30)]: 0
```

(The default device displayed may not be HD30, but you may ignore this discrepancy safely.)

Step 6 After copyright information is displayed, you are requested to enter one of the following options:

1. Full Restore
2. Partial Restore
3. Dual-Disk Upgrade
4. Release Upgrade

Since you are converting the system on your 3B2 Computer from a single-disk to a dual-disk system, enter option 3 in response to the following prompt:

```
Selection? [ 1 2 3 4 quit help ] 3
```

From this point, the procedure is interactive; the system prompts you through the Dual-Disk Upgrade procedure. You can enter **help** to get additional information.

Step 7 After the Dual-Disk Upgrade has completed successfully, the system automatically reboots from the hard disk, and prompts you with `Console Login:`.

Additional Installation Information

Misleading help Message During Release Upgrade

During the Release Upgrade, one **help** message lists the following features:

```
Upgrade features include:
            - Floating boot
            - Floating point assist in software support
            - XM (expansion module) software support
```

Floating boot is not provided by the system. Floating point assist and XM support are features that are provided by system software regardless of whether you install the system with the Release Upgrade or with some other procedure.

Not Having a Console

If you turn on a 3B2 Computer without having a terminal connected to the console port, initialization does not complete. When you connect a terminal, initialization completes.

Upgrading the Basic Networking Utilities

The Integral Modem package overwrites the **sysadm** menu from the Basic Networking Utilities package.

Re-install the Integral Modem after you have re-installed the Basic Networking Utilities. Otherwise, you cannot access the Integral Modem. If your Basic Networking Utilities package can access the Integral Modem, it cannot access STARLAN, and vice versa; the package cannot access both the Integral Modem and STARLAN.

You must install User Environment Utilities before you install the Basic Networking Utilities because the latter depends on **crontab**.

Insufficient Help Message During Full Restore

During a Full Restore, if you respond **help** to the following question, the message displayed does not state the sizes of the default partitions:

```
Use the default partitioning?
```

Refer to the "Hard Disk Default Partitions" section in Appendix A of the *System Administrator's Guide* for this information.

Messages About Partition Sizes May Be Wrong

When you do a Full Restore on a system with two disks and you do not choose the default partitioning, you get the following message:

```
How many blocks for the "usr" partition?
```

Suppose that you asked for **help** after this message. The text that appears after you ask for help contains one statement that may be wrong, and one that could be misleading.

In the following message, the *number* displayed with the message is incorrect when the two disks have different cylinder boundaries and you have **/usr** on the second disk:

`NOTE: partition sizes are always rounded up to cylinder` *number* `boundaries`

Nonetheless, partition sizes are rounded correctly, so ignore the number displayed.

The *number* displayed with the following message refers to the amount of space that **/usr** would take if you chose the default configuration, which puts **/usr** on the second disk.

```
The default "usr" partition size for your disk is number blocks
```

Incorrect Dual Disk Upgrade Message

When booting from the first core floppy, messages describing the available procedures are printed. The message describing the Dual-Disk Upgrade incorrectly states that it may be run on a 3B2 Computer Version 1 system or later releases. The Dual-Disk Upgrade may be performed only on a machine already running System V Release 3.0.

Rebooting the System Takes Time

When you reboot the system after Diagnostics passes, you may wait a long time for a prompt. This does not mean that the system is hung.

Ignorable Error Messages During Upgrade

If you have boards plugged into your 3B2 Computer, you see the following message during a Release Upgrade between the installation of the first and second core floppies while the system is rebooting:

```
SELF CHECK
UNKNOWN ID CODE 0X6 for device in SLOT 2
Equipt. dev. table complete will continue
Check EDT
```

Ignore these messages.

Unclear Error Message During Installation of lp Spooler

When you attempt to install the **lp** spooler package for the C-ITOH printer, the following error message is displayed:

```
Installation of printer dqp101 failed.  Save error messages.
```

Record the error messages showing which printer installation failed.

Existing Data Not Preserved When Installing lp Spooler

After installing the **lp** spooler package it seems that the existing printer data has been preserved because the command **lpstat −t** reports the expected printer names, but then the command while trying to get the 'accept' status of each.

Make sure existing data are saved before installing the **lp** spooler package.

Printer Uses Wrong Filter

The 5310 printer uses **/usr/bin/53filter** as the default filter even though this file does not exist. Use **/usr/lib/5310**.

DEbug MONitor (DEMON) PROMs

If you are running DEbug MONitor (DEMON) firmware, you may need to replace your PROMs when you upgrade your system to Release 3.0. To check whether you should replace your PROMs, follow these steps:

Step 1 Bring the machine to firmware mode.

Step 2 After entering the firmware password, type **boot**<CR>.

Step 3 When the machine requests the name of the program to **boot**, respond with **version**.

Step 4 The following information specifies PROMs that work with Release 3.0:

```
created: 5-31-85
release number: 1.3.0
load number: PF3
```

Step 5 If the date displayed is 7-24-84, you must replace your PROMs before attempting to use DEMON Software with Release 3.0. Call your account representative for details.

TTY Management Menu

The TTY Management Menu of the System Administration Menus has changed for UNIX System V Release 2.1 and Release 3.0. Please refer to the *Owner/Operator Manual* for details.

Installing the Cartridge Tape Utilities

If you have an AT&T 3B2 Computer without an XM, you cannot install the Cartridge Tape Utilities.

If you run **sysadm installpkg** to install the Cartridge Tape Utilities, the system specifically prompts you for the type of subdevices connected to the Cartridge Tape Controller (CTC) card. If you fail to respond to these prompts correctly, the system may prevent access to one or more of the subdevices.

If you wish to add an additional CTC card or CTC subdevice, you must remove the Cartridge Tape Utilities, install the card or device, and then reinstall the utilities so that all devices are recognized. See the *Cartridge Tape Utilities Guide* for details.

Installing the Terminal Information Utilities

The install script for the Terminal Information Utilities package (floppies 1 and 2) does not check if there is enough space for the package in the **root** (/) and **/usr** file systems before installing it.

The space requirements for this package are as follows. Floppy 1, which contains the **curses**(3X) library and various support utilities, must be completely installed, The first floppy requires about 150 blocks in the / file system and about 1000 blocks in the **/usr** file system. Floppy 2 contains the **terminfo**(4) database that you may install completely or partially (that is, selectively by terminal entry). The entire **terminfo**(4) database requires about 1300 blocks in the **/usr** file system. Each **terminfo**(4) entry requires, on the average, about 1 block.

Adding Printers to a Parallel Port

The script for adding printers to the lp system does not include a choice for a parallel port. Enter **/dev/tty15** for a parallel port on the first ports board, **/dev/tty25** for a parallel port on the second ports board, and so on, when this script prompts for the device name,

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.

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 power up 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.

- Login 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
```

file

For some files, the output of the **file** command is the following:

```
N/A on 3b2/300 w/paging (paged)
```

This means that the file is a UNIX System V Release 1.1 (or older) executable. You may execute files that provoke this response from **file** only on a 3B2/400 Computer equipped with a Math Accelerator Unit (MAU). Otherwise, you must recompile the program if you have the source code.

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 login 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 leftmost 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 leftmost 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**.

sar

The performance package does not report usage activity on devices which use the XDC driver. 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 pathname 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. a **stty** command with a carriage return select style argument (for example, **stty cr3**)
2. an **echo** command
3. a **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.

tee

In the following pipeline, the copies were not made on disks 2, 3, and 4.

```
dd if=/dev/rSA/diskette1 ¦
   tee /dev/rSA/diskette2 ¦
   tee /dev/rSA/diskette3 >
   /dev/rSA/diskette4
```

The problem is caused by the **tee** command writing in blocks of 16 bytes.

If you use the **block** device names (that is, **/dev/SA/diskette?**), the multiple copies succeed.

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) database) are, in general, between two and fourteen 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 multi-user mode, 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:

```
/usr/spool/uucppublic/receive/mth/my3b2/raleigh/files

/usr/spool/uucppublic/receive/mth/my3b2/durham/files
```

where **mth** is the destination user and **my3b2** is the name of your 3B2 Computer. Instead, **uuto** places all *files* from both directories under the following directory:

/usr/spool/uucppublic/receive/mth/my3b2/*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.

PATH

The default **PATH** environment variable searches the current directory first. A super-user unknowingly may run a program in the current directory.

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.

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.

NOTE 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.

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 Network 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
```

ctcinfo

The command **ctcinfo −r /dev/rSA/ctape** (which resets drive usage) is available to any user. To restrict use of the **ctcinfo** command (and its options) to super-users or **sysadm tapemgmt** users, the system administrator should use the following command:

```
chmod 550 /etc/ctcinfo
```

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. 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
```

errdump

When no errors have been logged, the **errdump** command does not cleanly handle it.

If messages show that **NVRAM** is invalid, or that panics occurred in January, 1970, it means that nothing has been logged.

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.*

fmtflop

If **fmtflop** is used without the −v option (verify), then the command completes but any writes to that floppy will result in I/O errors. If **fmtflop −v** is done, the verify phase will indicate errors. This problem is configuration dependent and will rarely occur. If you experience this problem, you can do the following:

1. Save the old **/etc/master.d/kernel**.
2. Save the old **/boot/KERNEL** someplace other than **/boot**.
3. Change the NPROC value in **/etc/master.d/kernel** by 38 (either increment or decrement).
4. Type the following commands:

   ```
   cd /boot
   mkboot -k KERNEL
   ```

5. Reboot from **/etc/system**

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, an **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 i-nodes, 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.

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.

prtconf

On rare occasions when the system has been powered up, **prtconf** does not display the XDC subdevices. This omission occurs because the XDC device has not spun up completely. Execute **sysadm reboot**.

The problem above also may occur when rebooting. If XDC subdevices are not displayed when rebooting, you have an old version of the firmware. Replace your firmware with the latest version.

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 autold

sysadm autold defines the device that the system uses to do an automatic load (auto load) if no other device is specified. By default, the auto load device is null. If the auto load default is not null and is not **/unix**, then a Release Upgrade, Partial Restore, or Full Restore fails. Also, if the default device number is 0 (floppy disk), these procedures fail.

Set the default auto load to **/unix**. Set the default device number to 1, which specifies to boot off the hard disk.

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.

NOTE 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 firmware

When you do a Partial Restore, files in **/etc/rc?.d** and **/etc/init.d** are copied to **/usr/old**. The original files are not removed. When the machine reboots after the Partial Restore finishes, the scripts in **/etc/rc?.d** are executed, and an error may occur because the system cannot find certain software (that software must be reinstalled).

This problem causes the following error message to appear after a Partial Restore on a system using STARLAN:

```
/usr/bin/nlsadmin: could not find correct password entry for listen
```

Ignore this message. Reinstalling the Networking Support Utilities product and STARLAN stops this message from reappearing. Adding additional files to the **/etc/rc?.d** directories may cause other problems.

sysadm portmgmt

You cannot login on the contty port if the computer is in **init** state 3 (for Remote File Sharing) or 4 (user defined). As root, edit the second field for contty in **/etc/inittab** to be **234** instead of the default state of **2**.

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 tapemgmt format

The **tapemgmt format** command allows the verify pass to be an option. The verify pass creates the defect map on the tape. If you do not run a verify pass, and a hard defect exists on the tape, you are backing up to a medium that contains defects. This causes the backups or restores to fail.

To get around this difficulty, follow these steps from the cartridge tape utilities exception letter:

Step 1 Reformat the tape using verify pass.

Step 2 Always use the verify pass during **format**, **backup**, and **ctccpio**.

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 end 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.

sysdef

sysdef does not display the following tunable parameters: SHLBMAX, GPGSMSK, PUTBUFSZ, and ILOGSIZE.

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.

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 **umounts**.

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. Login 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.

Path for Superuser Inconsistent

If you log in as **root**, the system default PATH contains the current directory. If you **su** to **root**, the system default PATH does not contain the current directory. Both PATH variables should be the same. The PATH should not contain the current directory.

Create a **.profile** for the **root** login that sets the PATH variable as follows:

```
PATH=/bin:/etc:/usr/bin
```

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**.

console

At times after rebooting the system, someone logging in on the console may receive the following message without being prompted for a password:

```
Login incorrect:
```

This may also occur if the superuser kills **hdelogger** while no one is logged into the console. The console becomes disassociated from **/dev/tty**. The process group of the console has become attached to one of the startup functions that execute out of **/etc/rc.d** or **/etc/inittab**.

Reboot the system.

console

When a hard-wired terminal is disconnected from the console port, and is reconnected, the console port echoes but does not respond or present a login prompt. Going to another terminal to investigate this problem shows no getty or shell process associated with the console. A getty runs on the console if you type **init q** or if **hdelogger** is killed. If you kill **hdelogger**, you should be wary of the problem discussed immediately above.

Kernel

When a corrupt file system is checked (for example, with **sysadm checkfsys**), the error message displayed differs depending on whether the file system resides on an integral or an XM floppy drive. The error message printed for the integral floppy drive is as follows:

```
NOTICE: Floppy Access Error: Consult the Error Message
Section of the System Administration Utilities Guide
```

The error message printed for the XM floppy drive is as follows:

```
NOTICE: CTC #: Access Error: Consult the Error Message
Section of the 3B2 Computer Cartridge Tape Utilities Guide
(error num=#)
```

On receipt of the integral floppy message, consult the "UNIX System NOTICE Messages" appendix of the *System Administrator's Guide*, not the obsolete *System Administration Utilities Guide.* On receipt of the XM floppy message, consult the error messages appendix of the guide named in the message.

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.

NOTE The default value for the **MAXUMEM** parameter is too large for most 3B2 Computer systems.

Change **MAXUMEM** in **/etc/master.d/kernel** to 1024, and then execute the following commands, after which `Console Login:` should appear:

```
# cd /boot
# mkboot -k KERNEL
# cd /
# sync
# sync
# touch /etc/system
# shutdown -i6 -g0 -y
```

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 **clists**, 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 **clists**.

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).

Kernel

Under heavy I/O, the system hangs after a panic.

When this happens, hit <RESET> and take a dump. The reset wipes out register contents, but panic saves the registers in NVRAM before the hang can occur.

Lost Characters on Input with 2x9600 with Flow Control

When using the ports cards in a configuration that is pumped by another machine, characters may be lost even with flow control turned on. When running two lines at 9600 baud, the ports card is able to keep up with or without flow control turned on. But if the machine is given a moderate load to keep it from devoting its attention to the ports, characters get lost. This would be expected in the raw/no flow control mode. It should not happen in the raw/flow control mode, but it does.

You may do one of or all the following things to overcome this problem: reduce your baud rate from 9600 to 4800 baud, buy more memory, or keep your system lightly loaded.

Manual Page for fs Format Is Incorrect

mount expects **s_magic** field of the given argument's superblock 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.

/etc/ports Does Not Insert gettys in /etc/inittab

After a partial restore, the new **/etc/inittab** does not contain **/etc/getty** entries for the tty lines on the ports board. When **/etc/ports** is run, these entries should be inserted into **/etc/inittab** after creating **/dev/tty??** entries. However, if **/dev/tty??** exists, **/etc/ports** does not insert the getty entries into **/etc/inittab**.

After a partial restore, if **/etc/inittab** does not contain getty entries for the tty lines on the ports board, you should remove **/dev/tty??** and run **/etc/ports** again.

NOTE

Do not remove **/dev/tty**.

/etc/ports then inserts getty entries into **/etc/inittab** and creates the appropriate **/dev/tty??** character special devices.

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.

DEMON Magic Mode

3B2 Computers equipped with DEbug MONitor (DEMON) Software PROMs have a feature known as "magic mode." The commands to enter magic mode are described in Chapter 3 of the *Debug Monitor Guide.*

When you initially enter the magic mode, some erroneous messages may be displayed. Ignore these messages. To prevent these messages from appearing, type **reset** when the firmware prompt appears, and then wait for the system to reset. Then you may enter magic mode in the usual way.

ctcinfo Error Message

The error message that occurs after you use **ctcinfo** on a non-formatted tape has changed. This is the new error message:

```
CTC X; Access Error: Consult the Error Message Section of the
3B2 Computer Cartridge Tape Utilities Guide(error num=215)
ctcinfo: cannot open /dev/rSA/ctapeX, errno = 215
```

(*X* is a number.)

Cipher Drive Operation Occasionally Misses End of Tape Markers

The internal Cipher drive operation of the Cartridge Tape Unit occasionally misses the end of tape (EOT) markers and causes the tape to run off the end. This should occur less than once in 300 cartridges.

If the operation misses EOT, and if you cannot recover data from any other backup, send the cartridge to your support representative.

Occasional Access Errors During Tape Reads

During tape accesses, the following error message may be printed:

```
CTC X; Access Error: Consult the Error Message Section of the
3B2 Computer Cartridge Tape Utilities Guide(error num=215)

            CTC X - Cartridge Tape could not
            read stream ?, segment ?, sector ?, status=0x??
            read stream ?, segment ?, sector ?, status=0x??
```

(*X* is a number.)

If you backup data onto tape, follow one or all the following steps:

Step 1 Remove cartridge tape, clean the heads, reinsert the tape, and re-execute the job.

Step 2 Reformat the cartridge tape. See the *Cartridge Tape Utilities Guide* for further information on how to format a cartridge tape.

Step 3 Remove cartridge tape and replace with another tape. The suspect tape should then be marked as having incurred an Access Error. Tapes that have recurring Access Errors should be disposed of properly.

If you are restoring data, execute the following commands:

Step 1 Unlatch and relatch tape drive, causing tape to be retensioned.

Step 2 If Step 1 does not work, remove cartridge tape, clean the heads, reinsert the tape, and re-execute the restore.

Possibility of Data Corruption After a Power Outage

Although the probability is low, the hard disk data may be severely corrupted after a power outage. If the data are severely corrupted, several warning messages are displayed when you turn on the computer. One of the messages is the following:

```
WARNING: The file system check has failed very
badly. We strongly suggest that you do not use
the machine until a service representative has
looked at it. I will now turn off power.
```

The system is then automatically powered off. Before you contact a service representative, try a full restore of the UNIX operating system using the 3B2 Computer core system floppy disks. A Full System Restore corrects the disk corruption, and the computer is again usable. See the *System Administrator's Guide* for the Full System Restore procedure.

NOTE A Full System Restore deletes all data on the disk unless you back them up.

File System Restore

Error Message When Restoring a Cartridge Tape File System

When **sysadm restore** fails on a cartridge tape file system, the following error message appears:

```
/etc/ctccpio: Tape read error.
The restore did not work.
Consult the Error Message Section of the
3B2 Computer Cartridge Tape Utilities Guide (error num=215)
```

Some of the corrupted files that were restored from the tape may have overwritten files on the hard disk. Repeat the restore procedure.

Error Messages During Complete Restore

During a "complete" restore of the **root** (/) and **/usr** file systems, messages similar to the following are displayed:

```
Cannot link </bin/sh> & </bin/rsh>
Cannot link </etc/init> & </etc/telinit>
Cannot unlink current </etc/cron> error 26 text file busy
Cannot unlink current </etc/getty> error 26
Cannot create </etc/getty> error 26 text file busy
Cannot unlink current </etc/hdelogger> error 26
Cannot create </etc/hdelogger> error 26
Cannot unlink current </usr/lbin/ncpio> error 26
Cannot create </usr/lbin/ncpio> error 26
```

It is normal to get these messages; they do not suggest a problem with the restore process.

Turn Off User Terminals During a Full System Restore or an Upgrade

During an upgrade or a full system restore, the only terminal that should be turned on is the console terminal. When an upgrade or full system restore is performed, all tunable parameters are set to default values for a 1M system. If several terminals are connected to Expanded I/O Capability feature cards and are turned on, the default parameter values may not be adequate to handle all the terminals. A possible consequence of this condition is that users may not be able to log in after a full system restore or an upgrade. To avoid problems, you need to re-tune the system parameters to their previous values before allowing users to turn on their terminals. A message is printed during an upgrade or restore process telling you to do this.

Change Access Permission After Full Restore on Two-Disk System

After doing a full system restore, the access permission on the **/usr** and **/usr2** file systems should be set to 775; that is, not everyone should have write permission. However, if you have a two-disk system and you do a full restore, the access permission on the **/usr2** file system is set to 777; that is, everyone has write permission. To change the access permission, log in as **root** and enter the following command:

```
chmod 775 /usr2
```

Reinstall Utilities With Software Drivers After a Partial Restore

NOTE

After doing a partial restore, any installed utilities that contain software drivers must be reinstalled.

See the *Owner/Operator Manual* to identify utilities that contain software drivers.

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 superuser, 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 lower-case 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
```

Reset Date If NVRAM Is Invalidated

If any information in NVRAM is changed, for example, when using the floppy key, the following message is displayed:

```
Time of Day Clock Needs Restoring:
Change using "sysadm datetime" utility
```

When the System Administration subcommand **sysadm datetime** is executed, the date and time are printed; and you are prompted to change them if they are not correct. Even if the time and date are correct, enter the time and date when you are prompted by the computer. Otherwise, the `Time of Day Clock Needs Restoring` message is printed every time you power up the computer.

Effect of Removing Power After NVRAM Sanity Failure

If an NVRAM Sanity Failure occurs and the 3B2 Computer is immediately powered down, the default boot device is changed to the floppy drive and all later boot attempts fail. To reset the default boot device back to the hard disk, follow the steps below:

Step 1 Power down the system.

Step 2 Insert the floppy key into the floppy drive.

Step 3 Power up the system. This automatically executes the floppy key and resets NVRAM to default values. The system then automatically boots the UNIX system from the hard disk.

Contents of sys crontab Is Wrong

/usr/spool/cron/crontab/sys is intended to contain performance collection commands. The file is installed by core floppy number 5, and should contain only a few comment lines. When the Performance Management Utilities package is installed, it edits **/usr/spool/cron/crontab/sys** to add the appropriate commands unless it finds that the commands already exist.

This crontab file installed by core floppy number 5 for UNIX System V Release 3 contains both the comment lines and the performance commands. Thus, between the time you install the core floppies and the time you install the performance package, cron sends mail to **/usr/spool/cron/crontab/sys** complaining that it cannot find **/usr/lib/sa/sa1** (one of the performance commands). It you decide not to install the Performance Management Utilities (possibly to save space on **/usr**), the file **/usr/mail/sys** grows by 5,424 characters per day, and the file **/usr/lib/cron/log** grows by an extra 2,520 characters per day. (avoid wasting space with mail messages)

Either install the Performance Management Utilities, or manually truncate **/usr/mail/sys** and **/usr/lib/cron/log**. If you do not install the performance package, remove non-comments from **/usr/spool/cron/crontab/sys** (the last three lines).

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.

Primary Server Panics

The primary server panics with the following scenario.

```
# init 3
# rfadmin -r secondary
FAILS -- can't remove secondary
# rtstop
TRAP
PANIC: KERNEL MMU FAULT (F_ACCESS)
```

A dump is available.

Take the system down and reboot.

Application Program Compatibility

The execution of some application packages may fail, and the following message may appear:

```
command: cannot execute
```

If this occurs, it means that if possible, you must make new object files with Issue 2 or later of the C Programming Language Utilities (CPLU).

To identify packages that may fail this way, use the following CPLU command with the executable file of the package (*filename*):

dump −**fv** *filename*

This produces a list of "flags." If the **BM32RST** flag is listed, the executable file was created using Issue 2 or later of the CPLU. Otherwise, you must generate a new object file.

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 filename 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 example below 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 Release 2.0

The sections that follow list the changes that affect compatibility with UNIX System V Release 2.0. Some of these changes became effective in System V Release 2.1, as you may know if you have already upgraded to Release 2.1. If you are upgrading directly to System V Release 3.0 from Release 2.0, you can refer to just this section to find all the changes that may affect you.

Each change listed below is marked with either (3.0) or (2.1) to distinguish in which release the change was made.

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

The figure below lists the UNIX System V commands affected by changes introduced in Release 2.1 or 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*
f77(1), **pc**(1)	*FORTRAN Programming Language 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 2: Summary of Changed Commands

cc(1) 2.1

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) 3.0

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 effect those who have the Security Administration Utilities package installed on their system.

dirname(1) 3.0

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) 3.0

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) 3.0

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.

NOTE 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) 3.0

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) 3.0

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) 3.0

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

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.

f77(1), pc(1) 3.0

> Summary of Change: The libraries linked with a program to be profiled are now in **/usr/lib** instead of **/lib**. This means that the **−p** option will no longer correctly arrange for a FORTRAN or Pascal program to be profiled.

While FORTRAN and Pascal programs can no longer be profiled by using the **−p** option as before, it is still possible to profile them by supplying more information to the command. In addition to the **−p** option, you must also include a direct reference to the files in the **/usr/lib/libp** directory.

If you are using a different language compiler that supports profiling, then you may have to include the **−L/usr/lib/libp** in the compilation step. If the **−L** option is not supported, the profiled libraries in the **/usr/lib/libp** directory must be explicitly named.

file(1) 3.0

> Summary of Change: Additions to the **file** command's repertoire of file types (found in the system file **/etc/magic**) make the command incompatible with an older **/etc/magic**.

You must use the **sysadm installpkg** command to ensure that the entire application package is installed. Moving only the new **file** command to an earlier release of UNIX System V, without moving the **/etc/magic** file as well, will cause the **file** command to fail.

NOTE

The old **file** command will work with the new **/etc/magic**, although no new features are gained by doing so.

fuser(1M) 3.0

> 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) 3.0

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

pr(1) 3.0

> 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) 3.0

> 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

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) 3.0

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) 3.0

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) 3.0

Summary of Change: The UNIX 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) 3.0

Summary of Change: The UNIX 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) 3.0

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) 3.0

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 2.1 or 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)
sys3b	**sys3b**(2)
umount	**umount**(2)
ustat	**ustat**(2)

Figure 3: Summary of Changed System Calls

acct(2) 3.0

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) 2.1

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 4: Deadlock Detecting System Calls

NOTE 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) 2.1

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) 2.1

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) 2.1

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) 2.1

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

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) 2.1, 3.0

> 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) 2.1

> 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) 3.0

> 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) 2.1

> 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) 3.0

> 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) 2.1

> 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) 2.1

> 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) 2.1

> 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) 2.1

> 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 looked for the **EINVAL** error return to indicate that the unlock attempt was not needed, it will no longer work as expected.

signal(2) 3.0

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) 3.0

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 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) 3.0

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.

sys3b(2) 2.1

Summary of Change: The **sys3b** system call, used with the **S3BSWAP** command, now allows the specification of additional swap devices. Previously the system call would fail on second and subsequent attempts.

Release 2.1 and Release 3.0 now allow more than one swap device. Although it is suggested that the newer **sys3b** command, **S3BSWPI,** be used instead, the less flexible **S3BSWAP** 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 change programs to use the new **S3BSWPI** command, for future compatibility.

sys3b(2) 3.0

Summary of Change: The **sys3b** system call, when used with the **S3BNVPRT, WNVR,** or **RNVR** commands, now fails, setting **errno** to **EFAULT,** if the non-volatile RAM (NVRAM) cannot be accessed. Also when used with the **WNVR** and **RNVR** commands the system call fails, setting **errno** to **EINVAL,** if an invalid NVRAM address is used. All of these are new checks for 3.0.

sys3b(2) 3.0

Summary of Change: The **sys3b** system call now sets **errno** to **EPERM** for all non-superuser errors.

Previously, when a regular user ran a program that called the **sys3b** 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 Note that if the superuser runs a program that calls **sys3b**, an incorrect use still causes **errno** to be set to EINVAL. Furthermore, this change only affects those **sys3b** uses that require superuser privilege.

umount(2) 3.0

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) 3.0

Summary of Change: For **umount**, **EBUSY** is now returned if the device of the filesystem 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 filesystem that can be unmounted (for example, **/usr**), and you attempt to unmount that filesystem, the unmount will fail with the above error.

ustat(2) 3.0

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 LibraryRoutines

The following figure lists the UNIX System V library routines affected by changes introduced in Release 2.1 or 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 5: Summary of Changed Library Routines

abort(3C) 3.0

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) 3.0

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) 3.0

> 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) 3.0

> 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) 3.0

> 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) 3.0

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, Release 2.1, and Release 3.0 documentation, but differs from the implementation on UNIX System V Release 2.0 and Release 2.1. 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) 3.0

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 2.1

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

Below are the error codes introduced in Release 2.1 and 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 6: Error Codes

The values of four error codes introduced in Release 2.1 were changed for Release 3.0. AT&T did not support the four error codes in Release 2.1, 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 2.1, 3.0

The figure below lists the header files changed, dropped, or introduced in Release 2.1 and 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 2.1. 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 2.1	Release 3.0
core.h	changed	
errno.h		changed
fcntl.h	changed	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	changed
* **sys/immu.h**	changed	
sys/inline.h		new
* **sys/inode.h**	changed	changed
sys/ipc.h	changed	
sys/lo.h		dropped
sys/map.h	changed	
sys/mau.h	changed	
* **sys/mount.h**	changed	changed
sys/nvram.h	changed	
sys/open.h	changed	
* **sys/param.h**	changed	changed
* **sys/pfdat.h**		new
* **sys/proc.h**	changed	
sys/psw.h	changed	
* **sys/reg.h**	changed	
sys/sbd.h	changed	
sys/shm.h	changed	
sys/stat.h		changed
* **sys/stream.h**		new
* **sys/stropts.h**		new
sys/sys3b.h	changed	
* **sys/sysinfo.h**	changed	changed

Header File	Release 2.1	Release 3.0
sys/sysmacros.h		changed
sys/systm.h	changed	changed
sys/tty.h	changed	
sys/types.h	changed	
* **sys/user.h**	changed	changed
* **sys/var.h**	changed	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 7: Changed, Dropped, or Added Header Files

NOTE

Because of the extensive changes made to the header files marked with an asterisk, 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 2.1

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 2.1, 3.0

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 6, Error Codes.

fcntl.h 2.1, 3.0

> 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 2.1

> 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. When set, this bit indicates that the Math Acceleration Unit (MAU) is required to run the program.

ldfcn.h 2.1

> 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 3.0

> 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 2.1

> 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 below:

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 3.0

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 3.0

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 3.0

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 3.0

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

sys/fcntl.h 3.0

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

sys/filsys.h 3.0

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 3.0

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 2.1, 3.0

Summary of Change: The contents of **sys/inode.h** have been changed extensively, and now contains 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 2.1

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 3.0

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

sys/map.h 2.1

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

sys/mau.h 2.1

Summary of Change: Several macros defining miscellaneous handling of the Math Acceleration Unit (MAU) and macros defining in-line assembly code have been added to **sys/mau.h.**

sys/open.h 2.1

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 2.1, 3.0

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/pcb.h 2.1

Summary of Change: The macros that define the kernel interrupt PSWs have been moved from **sys/pcb.h** to the **sys/psw.h** header file.

Any of your programs that use the **ZPSW** or **KPSW** *nn* macros for the kernel interrupt PSWs should now include the proper file with a line such as the following.

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

sys/proc.h 2.1, 3.0

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/psw.h 2.1

Summary of Change: The macros **PS_V** and **PS_Z** are now correctly defined; the definitions in earlier releases were reversed. Also, several definitions have been moved here from the **pcb.h** and **types.h** header files.

sys/sbd.h 2.1

Summary of Change: The default shared memory attach address, defined by the macro **UVSHM**, has changed.

sys/shm.h 2.1

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 to such 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 3.0

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/sys3b.h 2.1

Summary of Change: The **sys/sys3b.h** file now defines a new command, **S3BSWPI**, to use with the **sys3b** system call to declare a swap device. This command is more general than the old **S3BSWAP** command, although the old command is still supported.

sys/sysmacros.h 3.0

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 2.1, 3.0

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 2.1

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

sys/types.h 2.1

Summary of Change: The definition of the **psw** structure has been moved to the **sys/psw.h** header file, and the definition of the **sde_t**, **SRAMA**, and **SRAMB** types have been moved to the **sys/immu.h** header file. Also, the definitions for the **psram** and **mmuproc** structures have been removed.

sys/user.h 2.1, 3.0

Summary of Change: The **sys/user.h** header file has changed extensively, but the offsets of the following fields in the **user** structure have not changed.

u_base	**u_count**	**u_error**	**u_gid**
u_offset	**u_proc**	**u_qsav**	**u_r**
u_rgid	**u_ruid**	**u_segflg**	**u_ttyn**
u_uid			

sys/fs/s5dir.h 3.0
See the **sys/dir.h** section above for more information.

sys/fs/s5filsys.h 3.0
See the **sys/filsys.h** section above for more information.

sys/fs/s5param.h 3.0
See the **sys/param.h** section above for more information.

sys/fs/s5inode.h 3.0
See the **sys/inode.h** section above for information.

sys/fs/s5fblk.h 3.0
See the **sys/dir.h** section above for information.

AT&T 3B2 Miscellaneous Changes

brc entry in /etc/inittab 3.0

Summary of Change: The **brc** entry in the **/etc/inittab** file is now executed on the **bootwait** action, not the **sysinit** action as before.

This change should not have any affect on your use of the system.

Longest Allowed Pathnames 3.0

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

The length of a file's full pathname is now restricted to 1024 bytes. While previously the pathname 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 pathname. 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.

In any case, we encourage you to include the **limits.h** file with a statement like this

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

and refer to the **PATH_MAX** macro for the longest pathname allowed.

Changes Under Shared Libraries

The new shared libraries feature allows several processes to use routines in common without each having a separate copy of the code. However, you must recompile or relink a program to take advantage of this feature; The following sections describe some things to watch for. If you do not recompile or relink a program, then the shared libraries feature will not affect the program and it should continue to work as before.

Changes in Commands

chroot(1) 3.0

> 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 pathname 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) 3.0

Refer to the **chroot**(1) section above for changes to the **chroot**(2) system call.

exec(2) 3.0

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) 3.0

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 3B2 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	bug 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
ed	ed reports incorrect line number on "line too long"

Component	Description
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
graf	gtop does not work correctly on 3B2
graf	gtop output is missing data
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
mail	mail sometimes fails when sent via 3BNET
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

Component	Description
sh	shell does not recognize trailing colon as current directory in PATH.
sh	error in shell test.c code
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 rountine 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 3B2 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**
AT&T 3B2 Computer UNIX System V Release 3.0 Product Overview	305-556	**updated**
UNIX System V Release 3 Programmer's Reference Manual	307-226	**updated**
AT&T 3B2 Computer UNIX System V Release 3.0 Release Notes	305-557	**updated**
AT&T 3B2 Computer UNIX System V Release 3 System Administrator's Guide	305-558	**updated**
AT&T 3B2 Computer UNIX System V Release 3 System Administrator's Reference Manual	305-559	**updated**

Document Title	Select Code	Status
UNIX System V *User's Guide*	307–231	**updated**
UNIX System V *Release 3* *User's Reference Manual*	307-232	**updated**

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 *Primer*, provided with the optional Networking Support Utilities product, presents an overview of for programmers who will be using . Two other new networking support documents may also be purchased: the *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 Programmer's Guide	307-225	**new**
UNIX System V Release 3 STREAMS Primer (delivered with the optional Networking Support Utilities product)	307-229	**new**
UNIX System V Release 3 STREAMS Programmer's Guide (delivered with the optional Networking Support Utilities product)	307-227	**new**
UNIX System V Release 3 Network Programmer's Guide	307-230	**new**
AT&T 3B2 Computer UNIX System V Release 3 Networking Support Utilities Release 1.0 Release Notes (delivered with the optional Networking Support Utilities product)	307-233	**new**
AT&T 3B2 Computer UNIX System V Release 3 Remote File Sharing Utilities Release 1.0 Release Notes (delivered with the optional Remote File Sharing Utilities product)	307-224	**new**

C PROGRAMMING LANGUAGE UTILITIES ISSUE 4 RELEASE NOTES

Introduction

Overview

These *Release Notes* provide important information about Issue 4 of the C Programming Language Utilities (CPLU) for the AT&T 3B2 Computer. CPLU runs on any model of the 3B2 Computer, running UNIX System V Release 2.0 or later releases.

CPLU is a set of programming tools that help programmers develop C language programs. The main component is the compiler, called by the command **cc**. The **cc** command automatically calls the preprocessor, the assembler, and the link editor, as needed. Most programmers only need to access these tools through **cc** and do not need detailed information on the more intricate uses of a C compilation system.

These *Release Notes* contain essential information about CPLU, such as the installation procedure, plus descriptions of new software features. For more information on the use of a C compilation system, see the *UNIX System V Programmer's Guide*, which contains information for programmers who have never worked on a UNIX System before and for experienced programmers working on specialized applications who need to know the intricacies of a C compilation system. The *UNIX System V Programmer's Reference Manual* contains detailed reference material on the use of CPLU commands and C language functions.

For programmers who do advanced programming in the C language or who need to maintain different versions of files and programs, the Advanced Programming Utilities (APU) product contains extensive support tools. See the *C Programming Language Utilities Issue 4 and Advanced Programming Utilities Issue 1 Product Overview* for more information about APU.

Conventions Used in This Document

In this document, 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 exactly as they appear.
- *Italic* words are variables; you substitute the appropriate values. These values may be file names or they may be data values.
- CRT or terminal output and examples of source code are presented in `constant-width` font.
- In output and source code examples, a backslash (\) at the end of a line indicates that the line wraps around without a break.
- A command name followed by a number, for example, **prof**(1), refers you to that command's manual page, where the number refers to the section of the manual. These manual pages appear in the *Programmer's Reference Manual*, unless otherwise noted.

Contents of the Release

The C Programming Language Utilities come in these two sets of diskettes, the contents of which are displayed in the following tables:

- C Programming Language Utilities, Issue 4, on two diskettes
- Software Generation Utilities, Issue 4, on three diskettes

DIRECTORY	FILES			
/bin	cc	gencc	list	
/lib	comp	cpp		
/usr/include	a.out.h	ieeefp.h	poll.h	stropts.h
	aouthdr.h	ldfcn.h	prof.h	strselect.h
	ar.h	limits.h	pwd.h	syms.h
	assert.h	linenum.h	regexp.h	term.h
	core.h	macros.h	reloc.h	termio.h
	ctype.h	malloc.h	rje.h	time.h
	curses.h	math.h	scnhdr.h	tiuser.h
	dial.h	memory.h	search.h	tp_defs.h
	dirent.h	mnttab.h	setjmp.h	unctrl.h
	errno.h	mon.h	sgtty.h	unistd.h
	fatal.h	nan.h	signal.h	ustat.h
	fcntl.h	nlist.h	stand.h	utmp.h
	filehdr.h	nsaddr.h	stdio.h	values.h
	ftw.h	nserve.h	storclass.h	varargs.h
	grp.h			string.h
/usr/include/sys	system header files appropriate for your release of the UNIX operating system if you are running UNIX System V Release 2.1 or an earlier release.			
/usr/options	cc.name			

Figure 1: C Programming Language Utilities

DIRECTORY	FILES			
/bin	ar as conv	convert cprs dis	dump ld lorder	nm size strip
/lib	cm4defs crt0.o crt1.o crtn.o	fcrt0.o fcrt1.o fmcrt0.o fmcrt1.o	libc.a libc_s.a libld.a libm.a	libPW.a mcrt0.o mcrt1.o optim
/usr/bin	m4	tsort		
/usr/lib	libmalloc.a	llib-lmalloc.l	libcrypt.a	
/usr/lib/libp	libc.a	libm.a	libmalloc.a	
/usr/options	sgs.name			

Figure 2: Software Generation Utilities

New Software Features

This section summarizes features of CPLU Issue 4 that were not available with previous issues. Because many programmers do not need details about using these new features, this section provides brief, general descriptions of the new features' capabilities, with references to documents that contain more detailed information.

Shared Libraries Support

Issue 4 of the CPLU supports the use of shared libraries on UNIX System V Release 3.0. Shared libraries allow several **a.out** files to use the same object code simultaneously. Code compiled with CPLU Issue 4 can take advantage of shared libraries on the system. Shared libraries are described in detail in the "Shared Libraries" chapter of the *UNIX System V Programmer's Guide.*

Link Editor Enhancements

There have been several enhancements to the Link Editor Specification Language:

1. Two new output section types, **INFO** and **OVERLAY**, have been added.
2. An enhancement to the **SECTIONS** directive now allows the user to define how input sections should be combined to produce output sections. This enhancement enables the programmer who uses shared libraries to manipulate initialization sections as required.

These enhancements are described in detail in "The Link Editor" chapter of the *UNIX System V Programmer's Guide.*

New C Library Functions

The following paragraphs describe the many new C library functions in Issue 4 of CPLU.

There is a set of routines, which are described on the **directory**(3X) manual page, that programmers can use to access directories. These routines can only be used on UNIX System V Release 3.0 or later releases.

- **opendir** opens a specified directory and associates a directory stream with it.

- **closedir** closes the named directory stream.
- **readdir** returns a pointer to the next active directory entry.
- **seekdir** sets the position of the next **readdir** operation on the directory stream.
- **telldir** returns the current location associated with the named directory stream.

The two routines **mkdir** and **rmdir** are used to make a directory and remove a directory, respectively. These routines are only available on UNIX System V Release 3.0. They are described on the **mkdir**(2) and **rmdir**(2) manual pages.

The following routines have been implemented to support the STREAMS capability of UNIX System V Release 3.0.

- **getmsg**(2) is used to get a message from a STREAMS file
- **putmsg**(2) is used to send a message to a STREAMS file
- **poll**(2) is used for STREAMS input/output multiplexing

sigset, **sighold**, **sigignore**, **sigrelse**, and **sigpause** are new and provide signal management for applications processes These five routines are described on the **sigset**(2) manual page.

In addition, a new library routine, **dup2**, allows you to duplicate an open file descriptor. See **dup2**(3C) for details.

A new system call, **getdents**(2), reads directory entries and formats them as file systcm independent directory entries. **getdents** can only be used on UNIX System V Release 3.0 or later releases. See **getdents**(2) for details.

Proposed Standard for C

As these *Release Notes* were published, no official standard for the C programming language existed. The language accepted by AT&T C compilers follows the definition given in the *The C Programming Language* by B. Kernighan and D. Ritchie (Prentice-Hall, 1978). CPLU Issue 4 also supports some extensions:

- Flexnames – This extension allows variable and function name tokens to be distinct to at least the first 100 characters (rather than the first eight characters).

- Structure assignments and return values – This extension allows variables of the same structure type to be assigned to one another. The return value of functions can also be a structure.
- Enumeration types
- Multiple external variable declarations – This extension makes it possible to have the declaration

  ```
  int i;
  ```

  in multiple source files. All these multiple references resolve to the same address at link edit time.

Currently the X3/J11 task force of the American National Standards Institute (ANSI) is defining a standard for the C language (Preliminary Draft Standard, ANSI X3J11/85-102, August 11, 1985). The standard proposed by ANSI will allow most current legal C programs to be compiled without any changes. Nevertheless, to ease the possible transition process to the standard, the AT&T C compiler included with CPLU Issue 4 warns about uses of constructs that may not be legal in the future or may cause portability problems:

- Declarations, such as,

  ```
  int   i;
  static int i;
  ```

  produce the warning message

  ```
  warning: i previously declared extern, becomes static.
  ```

- Structure definitions missing semicolons, such as

  ```
  struct x {
    int  i
  }
  ```

  produce the warning message

  ```
  warning: syntax requires ; at end of struct/union decl
  ```

- The following code, which was not accepted by previous versions of the C compiler, is now legal:

```
static   f();
f(){
   ...
}
```

- The compiler supports single-precision arithmetic.

Obsolete Files

The following files are removed when you install Issue 4 of CPLU:

- **/lib/libp/libc.a**
- **/lib/libp/libm.a**
- **/lib/libp/libmalloc.a**

These header files, which existed in previous issues of CPLU, are obsolete and have been removed:

- **/usr/include/alarm.h**
- **/usr/include/dumprestor.h**
- **/usr/include/execargs.h**
- **/usr/include/symbol.h**

Installation Procedure

This section tells you how to use the System Administration menu command, **sysadm**, to install CPLU Issue 4 on your 3B2 Computer. Issue 4 of CPLU can be installed over a previous issue if one exists on your system.

Prerequisites

The following paragraphs describe CPLU storage requirements and software dependencies.

Software Dependencies

Before you can install and use CPLU, you must have installed the Directory and File Management Utilities. Also, if you are running UNIX System V Release 3.0, you must install the System Header Files that came with your operating system.

CPLU Issue 4 is supported on systems running UNIX System V Release 2.0 or later releases. The new shared library feature is supported only on systems running UNIX System V Release 3.0.

Storage Requirements

You must meet the following requirements before you begin installation.

- **Memory requirements**. The minimum memory requirement for CPLU is 420K of main memory.
- **Storage space**. There must be six megabytes of free disk storage. Installation will fail if there isn't adequate storage space. You can use the **df**(1M) command to check free disk storage.

  You need to have about 3100 blocks of free space in your root directory (**/**), and about 1300 blocks of free storage in **/usr**, plus room for System Header Files in **/usr**. The space needed to install System Header Files depends on what operating system you are running.

Shown below are the free blocks needed to install the System Header Files on different UNIX System V releases.

UNIX System V Release	**Blocks needed**
Before 2.1	**425 blocks**
2.1	**550 blocks**
3.0	**850 blocks**

CAUTION Don't install CPLU while any CPLU component is being used on the system.

Installing CPLU

1. Make sure **/usr** is mounted. Type **mount** to see what is mounted. If **/usr** has not been mounted, mount it with the **mount** command

   mount /dev/dsk/c*X*d*Y*s*Z* /usr

   where *X* is replaced by the controller number, *Y* is replaced by the drive number, and *Z* is replaced by the section or slice number where **/usr** is to be mounted.

2. Type the following command line:

   sysadm installpkg

   This executes the system administration menu subcommand **installpkg**.

3. Insert the first floppy diskette in the Software Generation Utilities set and press RETURN as instructed. After all the utilities on the first diskette have been installed, you will see a message that tells you to remove the first diskette and insert the next one. When you have repeated this procedure for all the diskettes in the package, you will see a message telling you to type **q** to signal the last diskette in the package.

   If there is not enough space in your root (/) directory to install the Software Generation Systems package, you will get a message telling you to either remove existing files from **/lib/libp** or move them into **/usr/tmp** so that the package can be installed.

CAUTION If you install Issue 4 of CPLU and later install AT&T's FORTRAN 1.0 or 1.1, Pascal, or BASIC 1.0, you should not install the Software Generation Utilities package included with those products. The version of the Software Generation Utilities package in Issue 4 of CPLU is the most recent and most efficient. It supports other AT&T language packages. If you install other versions over the Issue 4 version, you may get one of the following warnings when you try to use the **cc** command to produce object files:

```
aline 2: invalid instruction name
```

or

```
aline 2 : Invalid instruction name
aline 2 : syntax error
```

If you get one of these messages, reinstall the Software Generation Utilities package delivered with CPLU, Issue 4.

4. Repeat Step 3 for the C Programming Language Utilities package.

   If you are running UNIX System V Release 3.0, you will get a message that says you must install the System Header Files diskette in the Essential Utilities that came with your operating system package. You will get the message as you are installing the first diskette in the C Programming Language Utilities package. Even if you have installed System Header Files before, you must install them again to complete the installation so you can safely compile code. You should install the System Header Files after you've finished installing the second diskette in the C Programming Language Utilities package.

   If you are running UNIX System V Release 3.0, you must do the following procedure after installing CPLU Issue 4 to ensure correct floating point exception handling when you use the shared C library.

   As superuser, **cd** to **/lib** as shown:

   /bin/su
   cd /lib

Then type the following command line:

```
ar d /lib/libc_s.a  fpstart0.o
```

Maintaining CPLU Issue 4 and C-FP+

If you are maintaining the C-FP+ Programming Language Utilities along with CPLU Issue 4, and wish to take advantage the new system calls provided with UNIX System V Release 3.0, you should do the following after installing CPLU Issue 4.

- Become superuser by typing **/bin/su** and entering your root password.
- Change directory (**cd**) to **/tmp**.
- Type this command line:

```
ar x /lib/libc.a  signal.o
```

- Then type this command line:

```
ar r /usr/lib/fp/libc.a  signal.o
```

Software Notes

This section lists points of interest and workarounds that programmers might need to know about.

1. In early issues of some C compilation systems, all relocatable object files (**.o** files) produced by the assembler and relocated object files (**a.out** files) produced by the link editor had only three sections: **.text**, **.data**, and **.bss**. However, the assembler in Issue 4 can generate object files with an arbitrary number of sections in an arbitrary order; and the link editor can generate an arbitrary number of sections. This is because of the following features:

   - Addition of a **.comment** section (S-list) in most object files.
   - Elimination of zero-length **.bss** or **.data** sections in **.o** files. This change was introduced to enhance performance of the compilation process.
   - Addition of any number of user-defined sections for special-purpose applications, such as initialization code in some compilation systems.

   Some programs make assumptions about the number of sections in **.o** and **a.out** files. If you use Issue 4 to compile these programs, you should first change the programs so that they read the number of sections in the files. You can do this by reading in the file header using **ldfhread**(3X) and examining the **f_nscns** field of the file header. See **filehdr**(4) and **ldfhread**(3X) for details.

   If you are writing installable device drivers, you should keep in mind that the current versions of the **lboot** program (which makes a bootable UNIX System from the kernel and driver modules) assume that the installable drivers have three sections. You need to do two things to your driver's object files before running **lboot**:

   - Use the **mcs** command (in the Advanced Programming Utilities), as shown:

     mcs −d *drivername*.**o**

     This will delete the **.comment** section from the **.o** file.

- Add a **.bss** section to the **.o** file using the **ld** command, as shown:

  ld −r *drivername*.**o** **−o** *drivername*

  This takes *drivername*.**o** and produces the relocatable object *drivername*, attaching an empty **.bss** section to the input file.

  If you are using Basic 1.0 and linking **.o** files created from C source files, you should follow this same procedure.

2. On systems running UNIX System V Release 2.1 or earlier releases, the **/usr/include/sys** headers duplicated in CPLU may not work correctly if some AT&T software packages, namely the Kernel Source, Inter-Process Communications, and C Programming Language Utilities are installed and then removed from the hard disk. It is recommended that once the above utilities are installed, they remain on the hard disk and not be removed.

3. When you try to compile a file that resides on a file system that is mounted read-only (a diskette for example), the following error message is generated:

   ```
   assembler: file.c aline ###(cline #)
   Cannot open Output File *** error code 1
   ```

   The compilation will fail even if the output is directed to a read/write file system. The file system should either be remounted read/write (simple administration default), or the source file should be copied to a read/write file system (**/usr** for example), before compilation.

4. On UNIX System V Release 2.0 or earlier systems, the optimizer can run out of memory for very large functions. If it runs out of data space, it fails with the message:

   ```
   optimizer failed : no space available
   The -O option to cc is ignored.
   ```

   However, if it runs out of stack space, it aborts with the diagnostic:

   ```
   fatal error in optim.
   error code 0213
   ```

There are two ways around this problem.

- Compile without the **−O** option.
- Increase the **MAXUMEM/MAXMEM** parameter in the kernel so that bigger processes can run. (See the *System Administration Utilities Guide* for how to increase **MAXUMEM/MAXMEM**.)

5. **ld** −N produces a.outs that cannot be executed.

6. The following table lists argument/return value types that have changed. (In the second column, the entry "arg2" means "the second argument to the function," "arg3" means "the third argument," etc.)

Function Name	**Argument**	**Changed From**	**-->**	**To**
fread	**arg2**	**int**	**-->**	**size_t**
fwrite	**arg2**	**int**	**-->**	**size_t**
strncat	**arg3**	**int**	**-->**	**size_t**
strncmp	**arg3**	**int**	**-->**	**size_t**
strncpy	**arg3**	**int**	**-->**	**size_t**
ctime	**arg1**	**long**	**-->**	**time_t**
localtime	**arg1**	**long**	**-->**	**time_t**
gmtime	**arg1**	**long**	**-->**	**time_t**

For definitions of the new return value types, use

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

7. Profiled libraries have been moved from **/lib/libp** to **/usr/lib/libp**. Some language products (FORTRAN-77 1.1 or Pascal, for example) will not find them unless users type −L **/usr/lib/libp** on their command lines.

8. The C preprocessor **cpp** now warns about ignored tokens following directives. For example:

```
#ifdef a | b            /* | b not allowed */
#else  u3b2             /* u3b2 not allowed */
#endif u3b2             /* u3b2 not allowed */
```

9. The **size** command produces poorly formatted output on some object files (**.o** files). In most cases, the output is still understandable. For example, the output might have a trailing extraneous plus (+).

10. Most functions defined within the shared C library may be redefined by a user program. If users define their own versions of **malloc**() and **free**(), they must define the function (possibly null) **realloc**() to avoid multiple definitions of **malloc** and **free**.

11. FORTRAN programs that write to a file before reading it (for example, programs which prompt the user) will not execute as expected. This is because the output buffer won't be flushed before the read, due to changes in the **setvbuf**(3X) function. To work around this problem, you can extract and use with FORTRAN programs the earlier version of **setvbuf** included in the second Software Generation Utilities floppy in FORTRAN-77 Release 1.1. After the FORTRAN Programming Language Utilities are installed, you can follow these steps.

    - Become superuser (**su** command).
    - Check that the directory **/mnt** exists and create it if it does not.
    - Put the second floppy from the Software Generation Utilities that come with FORTRAN-77 Release 1.1 in the system driver and type

      ```
      mount /dev/diskette  /mnt -r
      ```
    - Change directory to **/tmp** (**cd**).
    - Extract **setvbuf.o** from **libc.a** on the floppy by typing

      ```
      ar x /mnt/new/lib/libc.a setvbuf.o
      ```
    - Put **setvbuf.o** at the end of **libI77.a** (the FORTRAN I/O library) by typing

      ```
      ar q /usr/lib/libI77.a setvbuf.o
      ```
    - Unmount the floppy by typing

      ```
      umount /dev/diskette
      ```

Compatibility

There are three general changes that occur in UNIX System V Release 3.0 that might affect the compatibility of your C language programs if they are intended to run on different releases of UNIX System V.

1. the recognition of a new C preprocessor directive, #ident, and the addition of it to UNIX System V Release 3.0 system header files
2. the addition of new error codes and system calls returning new or different error codes
3. changes to UNIX System V Release 3.0 system header files

For detailed information about these issues of compatibility among different releases of UNIX System V and how they might affect your programs, see the *UNIX System V Release 3.0 Release Notes* (select code 305-557). For information on ordering, see "How To Order Documents".

Documentation

This section describes the documents included with CPLU and tells you how to order additional copies and optional documents.

Document Descriptions

The following five documents are included with Issue 4 of CPLU:

1. The *C Programming Language Utilities Issue 4 and Advanced Programming Utilities Issue 1 Product Overview* (select code 307-182) contains a summary of the C Programming Language Utilities Issue 4 and the Advanced Programming Utilities Issue 1. The *Product Overview* is especially useful for new users.
2. The *C Programming Language Utilities Release Notes* (this document) contains the CPLU installation procedure, release information, and descriptions of new features, with references to relevant sections in the other documents that come with CPLU.
3. The *UNIX System V Programmer's Guide* (select code 307-225) contains descriptive information about programming on a UNIX System, the C language and associated libraries, the C compiler, tools such as **sdb**, **yacc**, **lex**, and **make**, shared libraries, SCCS (Source Code Control System), the Link Editor Specification Language, and much more.
4. The *UNIX System V Programmer's Reference Manual* (select code 307-226) contains reference material in the form of manual pages for programming commands, system calls, subroutines, libraries, file formats, macro packages, and character-set tables.
5. The *C Programmer's Handbook* (select code 307-135) contains reference material for the C language. Topics covered include syntax, data types, operators and expressions, statements, functions, declarations, program structure, libraries, formatted input/output, and portable C programs.

How to Order Documents

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

1-800-432-6600 (toll free within the continental United States)
1-317-352-8556 (outside the continental United States)

or by writing to:

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

ADVANCED PROGRAMMING UTILITIES ISSUE 1 RELEASE NOTES

Introduction

Overview

These *Release Notes* contain information about the Advanced Programming Utilities (APU). APU is a set of tools that are useful to programmers who

- do extensive programming in the C language,
- need tools to do advanced programming and symbolic debugging,
- want to create shared libraries,
- or work in an environment where it is necessary to track and maintain versions of files and programs.

APU includes the following packages:

- Advanced C Utilities, containing tools such as **cxref**, **ctrace**, **cflow**, and **lint** for the C language programmer, plus libraries, and **mkshlib** (to create shared libraries)
- Extended Software Generation Utilities, containing tools such as **m4** (a macro processor), **yacc** (Yet Another Compiler-Compiler), **sdb** (a symbolic debugger), **lex** (a generator of lexical analyzers), and **make** (a program construction tool)
- Source Code Control Utilities (SCCS), a system used to track changes made to files and to maintain a record of all versions

APU runs on any model of the AT&T 3B2 Computer running UNIX System V Release 2.0 or later releases.

These *Release Notes* contain the installation procedure for APU, a description of available documentation, technical information, and a description of new features.

Conventions Used in This Document

In this document, 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 exactly as they appear.
- *Italic* words are variables; you substitute the appropriate values. These values may be file names or they may be data values.
- CRT or terminal output and examples of source code are presented in `constant-width` font.
- In output and source code examples, a backslash (\) at the end of a line indicates that the line wraps around without a break.
- A command name followed by a number, for example, **prof**(1), refers you to that command's manual page, where the number refers to the section of the manual. These manual pages appear in the *Programmer's Reference Manual,* unless otherwise noted.

Contents of the Release

APU comes on three diskettes:

- Advanced C Utilities, Issue 4, on 1 diskette
- Extended Software Generation Utilities, Issue 4, on 1 diskette
- Source Code Control Utilities, on 1 diskette

The directory structure and files are presented in the following tables.

DIRECTORY	FILES			
/bin	**mkshlib**			
/usr/bin	**cb** **cflow**	**ctc** **ctcr**	**ctrace** **cxref**	**lint** **regcmp**
/usr/lib	**dag** **flip** **lint1** **lint2**	**llib-lc** **llib-lc.ln** **llib-lm** **llib-lm.ln**	**llib-port** **llib-port.ln** **lpfx** **nmf**	**xcpp** **xpass**
/usr/lib/ctrace	**runtime.c**			
/usr/options	**acu.name**			

Figure 1: Advanced C Utilities

DIRECTORY	FILES		
/usr/bin	**lex** **mcs**	**prof** **sdb**	**yacc**
/usr/lib	**libg.a** **libl.a**	**liby.a** **sdbp**	**sdbs** **yaccpar**
/usr/lib/lex	**ncform**	**nrform**	
/usr/options	**esg.name**		
/bin	**make**		
/etc	**install**		

Figure 2: Extended Software Generation Utilities

DIRECTORY	FILES			
/usr/bin	**admin** **cdc** **comb** **delta**	**get** **prs** **rmdel** **sact**	**sccsdiff** **unget** **val** **vc**	**what**
/usr/lib/help	**ad** **bd** **cb** **cm**	**cmds** **co** **de** **default**	**ge** **he** **prs** **rc**	**un** **ut** **vc** **lib/help** **lib/help2**
/usr/options	**sccs.name**			

Figure 3: Source Code Control Utilities

Software Features

The following paragraphs contain brief descriptions of the features and and some of the commands in this issue of APU. You may be familiar with some of these features, while others may be new to you. Even though this is the first release in which all of these features are part of the same product, most of these features have been available in other software packages.

Advanced C Utilities

Below are some of the Advanced C Utilities and their functions:

cxref	is a C cross-reference listing generator
ctrace	is a statement-by-statement execution trace facility
cflow	produces a graph of program dependencies
lint	detects faulty and non-portable code.
cb	displays the structure of code
regcmp	compiles regular expressions

All of these tools are described in the *UNIX System V Programmer's Guide* and the *UNIX System V Programmer's Reference Manual.*

The Advanced C Utilities package also contains **mkshlib**(1) (make shared library), which is used to create a shared library. Shared libraries are a feature of UNIX System V Release 3.0 that allow several **a.out** files to simultaneously use the same object code. The **mkshlib** command has options that allow you to specify the shared library specification file (which contains all the information necessary to build the shared library) and to name the host and target shared libraries. **mkshlib** and the shared library feature are described in detail in the "Shared Libraries" chapter of the *UNIX System V Programmer's Guide.*

Shared Library Upward Compatibility

Shared library compatibility is an important issue. These paragraphs explain how to build upward-compatible shared libraries. For more detailed information, see the "Shared Libraries" chapter in the *UNIX System V Programmer's Guide.*

Comparing Previous Versions of the Library

Shared library developers normally want newer versions of a library to be compatible with previous ones. **a.out** files will not execute properly otherwise. There are procedures that let you check libraries for compatibility. In these tests, two libraries are said to be compatible if their exported symbols have the same addresses.

To compare two target shared libraries, we look at their symbols and delete everything except external symbols. Then we create lists of symbol names and values for the new and old libraries, and compare the symbol values to identify differences.

If all symbols in the two libraries have the same values, the libraries are compatible. If some symbols are different, the two libraries may be incompatible. The procedure for comparing shared libraries outlined above is explained in detail in the "Shared Libraries" chapter of the *UNIX System V Programmer's Guide.*

Dealing With Incompatible Libraries

When you determine that two libraries are incompatible, you have to deal with the incompatibility. You can rebuild all the **a.out** files that use your library, or you can give a different target path name to the new version of the library. The host and target path names are independent, so you don't have to change the host library path name. New **a.out** files will use your new target library, but old **a.out** files will continue to access the old library.

NOTE

You should try to avoid multiple library versions. If too many copies of the same shared library exist, they might actually use more disk space and more memory than the equivalent relocatable version would have.

Extended Software Generation Utilities

The following list describes some of the tools in the Extended Software Generation Utilities package:

- **mcs**(1) is used to manipulate the **.comment** sections in object files. (**.comment** sections are created by **#ident**.) **mcs** can be used to delete, print, compress, or add to **.comment** sections.
- The symbolic debugger, **sdb**(1), is used to examine C language executable files and core files and provides a controlled environment for their execution. When testing C language programs symbolically, breakpoints can be set at executable lines of the source code. These breakpoints force the program to pause at the specified point so that an inspection can be made of the current state of the program.
- The **make**(1) program helps users build and maintain up-to-date versions of programs. **make** simplifies the job of keeping track of which files depend on other files, recently modified files, files that need recompiling after changes, and the sequence of operations needed to make a new version of a program.
- **lex**(1) generates programs to be used in simple lexical analysis of text. **lex** reads a file containing specifications of strings to be matched and associated C code. Whenever the lexical analyzer produced by **lex** matches a specified string in its input, it executes the associated C code.
- **yacc**(1) (Yet Another Compiler-Compiler) is a software tool that accepts an LALR(1) grammar specification and associated C code fragments that represent actions to be taken when a found grammar rule is reduced.

For more information about these commands, see the *UNIX System V Programmer's Guide* and the *UNIX System V Programmer's Reference Manual.*

Source Code Control Utilities

The Source Code Control System (SCCS) can be used to record all enhancements and changes to files, along with comments on each version, to maintain a history of the changes made. Some SCCS functions are

- retrieving any recorded version of a file with comments,
- storing a new version of a file,
- and comparing two versions of an SCCS file.

SCCS takes custody of a file and, when changes are made, identifies and stores them in the file with the original source code and/or documentation. As other changes are made, they too are identified and retained in the file. Each separate set of changes is called a delta. History data can be stored with each version: why the changes were made, who made them, when they were made.

Retrieval of the original or any set of changes is possible. Any version of the file as it develops can be reconstructed for inspection or additional modification.

SCCS Commands

Here is a list of SCCS commands:

get	retrieves versions of SCCS files
unget	undoes the effect of a **get –e** prior to the file being delta'd
delta	applies deltas (changes) to SCCS files and creates new versions
admin	initializes SCCS files, manipulates their descriptive text, and controls delta creation rights
prs	prints portions of an SCCS file in user specified format
sact	prints information about files that are currently out for edit
help	gives explanations of error messages
rmdel	removes a delta from an SCCS file. Allows removal of deltas created by mistake

cdc	changes the commentary associated with a delta
what	searches any UNIX System file(s) for all occurrences of a special pattern and prints out what follows it. Useful in finding identifying information inserted by the **get** command
sccsdiff	shows differences between any two versions of an SCCS file
comb	combines consecutive deltas into one to reduce the size of an SCCS file
val	validates an SCCS file
vc	a filter that may be used for version control

For instructions on how to use SCCS and detailed descriptions of SCCS commands, see the "Source Code Control System" chapter in the *UNIX System V Programmer's Guide.*

Software Installation Information

You will use the System Administration menu command, **sysadm**, to install the Advanced Programming Utilities on your 3B2 Computer.

Prerequisites

The following paragraphs describe CPLU storage requirements and software dependencies.

Software Dependencies

Before you can install and use APU, you must have installed the Directory and File Management Utilities. Also, if your operating system is UNIX System V Release 3.0 or a later release, you must have installed the System Header Files that came with your operating system.

Issue 1 of APU will be supported on systems running UNIX System V Release 2.0 or later releases. However, the **mkshlib** command will only work on UNIX System V Release 3.0 and later releases, which support the shared library feature.

Storage Requirements

You must meet the following requirements before you begin installation.

- **Memory requirements**. The minimum memory requirement for the APU is 420K of main memory.
- **Storage space**. There must be six megabytes of free disk storage. Installation will fail if there isn't adequate storage space. You can use the **df**(1M) command to check free disk storage.

  You need to have about 250 blocks of free space in your root directory (/), and about 3000 blocks of free storage in **/usr**.

Installing APU

1. Make sure **/usr** is mounted. Type **mount** to see what is mounted. If **/usr** has not been mounted, mount it with the **mount** command

   mount /dev/dsk/c*X*d*Y*s*Z* /usr

   where *X* is replaced by the controller number, *Y* is replaced by the drive number, and *Z* is replaced by the section or slice number where **/usr** is to be mounted.

2. Type the following command line:

   sysadm installpkg

   This executes the system administration subcommand **installpkg**.

3. Insert the first floppy diskette in the Extended Software Generation Utilities set and press RETURN as instructed. After all the utilities on the first diskette have been installed, you will see a message that tells you to remove the first diskette and insert the next one. When you have repeated this procedure for all the diskettes in the package, you will see a message telling you to type **q** to signal the last diskette in the package.

4. Repeat the procedure for the Advanced C Utilities package and, finally, the Source Code Control Utilities package.

Software Notes

This section lists points of interest and workarounds that programmers might need to know about.

1. Functions that use floating point may not be placed in a user-defined shared library. Applications that build their own shared libraries must arrange to place floating point code in a non-shared portion of the host archive shared library.

2. The command **mcs –d** will corrupt a.outs and object files where the comment section is not the last section. Use **mcs –d –ax** instead.

3. When compiling C programs that are the output of **ctrace**, expect to see warning messages of the form:

```
"/usr/lib/ctrace/runtime.c", line nnn warning:
illegal pointer combination, op =
```

4. The following C library functions do not have **lint** library definitions:

mkdir()	**opendir()**	**fpsetmask()**
rmdir()	**readdir()**	**fpsetround()**
sigset()	**closedir()**	**fpsetsticky()**
sighold()	**telldir()**	**isnand()**
sigignore()	**seekdir()**	
sigrelse()		**getutent()**
sigpause()	**lockf()**	**getutid()**
getmsg()	**cfree()**	**getutline()**
putmsg()		**pututline()**
poll()	**fpgetmask()**	**endutent()**
dup2()	**fpgetround()**	**setutent()**
getdents()	**fpgetsticky()**	**utmpname()**

5. The following C library functions have incorrect **lint** library definitions:

   setvbuf
   signal

6. The files **/usr/options/acu.name** and **/bin/mkshlib** are not removed when the Advanced C Utilities diskette is un-installed.

7. On UNIX System V Release 2.0 systems, **sdb** might look as though it has failed at startup. This is due to the kernel sending a signal to the process when it queries the kernel about its floating point capability. **sdb** reports something like:

   ```
   Bad System Call (12) (sig 12)
    at
   fpstart1.c: No such file or directory
   0x808????? in sys3b:No lines in file
   ```

   When this message appears, type **c** to continue.

8. In early issues of some C compilation systems, all relocatable object files (**.o** files) produced by the assembler and relocated object files (**a.out** files) produced by the link editor had only three sections: **.text**, **.data**, and **.bss**. However, the assembler in Issue 4 of the C Programming Language Utilities can generate object files with an arbitrary number of sections in an arbitrary order; and the link editor can generate an arbitrary number of sections. This is because of the following features:

   - Addition of a **.comment** section in most object files. (See "**#ident** Preprocessor Directives" in this document.)
   - Elimination of zero-length **.bss** or **.data** sections in **.o** files. This change was introduced to enhance performance of the compilation process.
   - Addition of any number of user-defined sections for special-purpose applications, such as initialization code in some compilation systems.

   Some programs make assumptions about the number of sections in **.o** and **a.out** files. If you use Issue 4 of the C Programming Language Utilities to compile these programs, you should first change the programs so that they read the number of sections in the files. You can do this by reading in the file header using **ldfhread**(3X) and examining the **f_nscns** field of the file header. See **filehdr**(4) and **ldfhread**(3X) for details.

For example, if you are writing installable device drivers, you should keep in mind that some versions of the **lboot** program (which makes a bootable UNIX System from the kernel and driver modules) assume that the installable drivers have three sections. You need to do two things to your driver's object files before running **lboot**:

- Use the **mcs**(1) command, as shown:

  ```
  mcs -d drivername.o
  ```

  This will delete the **.comment** section from the **.o** file.

- Add a **.bss** section to the **.o** file using the **ld** command, as shown:

  ```
  ld -r drivername.o -o drivername
  ```

This takes *drivername*.**o** and produces the relocatable object *drivername*, attaching an empty **.bss** section to the input file.

If you are using Basic 1.0 and linking **.o** files created from C source files, you should follow this same procedure.

9. You should not use **sdb** to debug any process which uses shared libraries.

10. The following table lists argument/return value types that have changed. (In the second column, the entry "arg2" means "the second argument to the function," "arg3" means "the third argument," etc.)

Function Name	Argument	Changed From	-->	To
fread	arg2	int	-->	size_t
fwrite	arg2	int	-->	size_t
strncat	arg3	int	-->	size_t
strncmp	arg3	int	-->	size_t
strncpy	arg3	int	-->	size_t
ctime	arg1	long	-->	time_t
localtime	arg1	long	-->	time_t
gmtime	arg1	long	-->	time_t

For definitions of the new argument value types, use

#include < sys/types.h >

11. The **mkshlib** command does not accept full pathnames for the **-h** options. It assumes the current directory and prepends the current working directory to the modifier of **-h**.

Documentation

These *APU Release Notes* (select code 307-184) come with APU. The *Release Notes* contain a description of APU and its main features, installation information, prerequisites, and storage requirements.

Related Documents

The following documents contain more information about features of APU and can be ordered as described in the next section.

1. The *C Programming Language Utilities Issue 4 and Advanced Programming Utilities Issue 1 Product Overview* (select code 307-182) contains a brief technical description of the C Programming Language Utilities, Issue 4, and the Advanced Programming Utilities Issue 1. The *Product Overview* is especially useful for new users.
2. The *UNIX System V Programmer's Guide* (select code 307-225) contains descriptive information about SCCS (Source Code Control System), the Link Editor Specification Language, **yacc**, **lex**, **make**, the symbolic debugging program **sdb**, shared libraries, programming on a UNIX System, the C language and associated libraries, the C compiler, and much more.
3. The *UNIX System V Programmer's Reference Manual* (select code 307-226) contains reference material in the form of manual pages for programming commands, system calls, subroutines, libraries, file formats, macro packages, and character-set tables.
4. The *C Programmer's Handbook* (select code 307-135) contains reference material for the C language. Topics covered include syntax, data types, operators and expressions, statements, functions, declarations, program structure, libraries, formatted input/output, and portable C programs.

How to Order Documents

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

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

1-317-352-8556 (outside the continental United States)

or by writing to:

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

NETWORKING SUPPORT UTILITIES RELEASE 1.0 RELEASE NOTES

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 login 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**) to be 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 Organization for Standardization (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 on one floppy diskette and requires approximately 440 free blocks in **/usr** and 651 free blocks in /.

NOTE 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:

/boot/log.o
/boot/clone.o
/boot/timod.o
/boot/tirdwr.o
/etc/master.d/clone
/etc/master.d/log
/etc/master.d/timod
/etc/master.d/tirdwr
/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

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 3B2/300, 3B2/310, and 3B2/400 Computers. The minimum hardware configuration required is 2 megabytes of main memory.

Installation Procedure

The following describes the installation procedures for the Networking Support Utilities. The Networking Support Utilities for the 3B2 Computer are distributed on one floppy diskette. Most of the utilities are object code files. To begin the installation:

- Login as the superuser and be sure you are in the root (/) directory.
- Place your computer in system state–2 (multi-user) 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 and, when prompted, press **<CR>** as instructed. Once you have hit **<CR>**, your 3B2 Computer will display the full path names of the files as they are copied from the floppy diskette to the hard disk. Be patient, this will take several minutes.

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

Step 4: As instructed, you should shutdown the system and bring it back up again. This is done as follows:

```
# cd /
# shutdown -i6 -g0 -y
```

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 database 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 > /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 newline 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 database 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:

```
Called 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**.

Documentation

The following documents are provided with the Networking Support Utilities:

AT&T 3B2 Computer Networking Support Utilities Release 1.0 Release Notes (select code 307-233)

STREAMS Primer (select code 307-229)

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

STREAMS Programmer's Guide (select code 307-223)

Network Programmer's Guide (select code 307-230)

For further information on additional documentation for System V Release 3, see the *Product Overview* (select code 305-556) 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 AT&T Customer Information Center (CIC):

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

1-317-352-8556 (outside the continental United States)

or by writing to:

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

REMOTE FILE SHARING UTILITIES RELEASE 1.0 RELEASE NOTES

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.

/boot/du.o
/boot/dufst.o
/boot/sp.o
/etc/rmount
/etc/rmountall
/etc/rumountall
/etc/master.d/du
/etc/master.d/dufst
/etc/master.d/sp
/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

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 3B2/300, 3B2/310, and 3B2/400 Computers. 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

This procedure describes Remote File Sharing installation. The Remote File Sharing Utilities for the 3B2 Computer are distributed on one floppy diskette. 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 (multi-user) 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.
- Login 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 and press **RETURN** as instructed. Once you have hit **RETURN**, your 3B2 Computer will display the full path names of the files as they are copied from the floppy diskette to the hard disk. Be patient, this will take several minutes.

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

dork.371

Step 4: As instructed, you should shutdown the system and bring it back up again. This is done as follows:

```
# shutdown –i6 –g0 –y
```

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 pathname, **acct** will return an **errno** of **EIN-VAL**.

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 filesystem (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 filesystem, 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 non-numeric 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 the 3B2 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 **2:**), 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 pathname 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 **/etc/master.d/kernel**. 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 **/etc/master.d/du**. 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 re-running 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
```

Documentation

The two documents you will need to set up and run Remote File Sharing are:

- *AT&T 3B2 Computer UNIX System V Release 3.0 System Administrator's Guide* (select code 305-558).
- *AT&T 3B2 Computer UNIX System V Release 3.0 System Administrator's Reference Manual* (select code 305-559).

You will also need AT&T STARLAN NETWORK documentation for information on installing and maintaining the STARLAN NETWORK.

- *STARLAN NETWORK AT&T 3B2 Computer Network Program Installation Guide.*
- *STARLAN NETWORK AT&T 3B2 Computer Network Program User's Guide.*

For further information on additional documentation for System V Release 3, see the *Product Overview* (select code 305-556) 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 AT&T Customer Information Center (CIC):

1-800-432-6600 (toll free within the continental United States)
1-317-352-8556 (outside the continental United States)

or by writing to:

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

PRENTICE-HALL, INC., Englewood Cliffs, N.J. 07632

ISBN 0-13-940552-6