INTERACTIVE™ UNIX® System V/386 Release 3.2 Operating System Guide

INTERACTIVE

p r o d u c t   f a m i l y

UNIX is a registered trademark of UNIX System Laboratories, Inc. in the United States and other countries.

SunSoft
A Sun Microsystems, Inc. Business

© 1992 Sun Microsystems, Inc.—Printed in the United States of America.
6601 Center Drive West, Los Angeles, California 90045 U.S.A.

© 1987-1988 AT&T Corporation

© 1981, 1982, 1983, 1984, 1985, 1986, 1987 Microsoft Corporation

© 1984, 1985, 1986, 1987 Phoenix Technologies Ltd. and Sun Microsystems, Inc.

© 1980, 1983 Regents of the University of California

© 1983 Eric P. Allman, Berkeley, CA

© 1988 The Institute of Electrical and Electronics Engineers, Inc.

All rights reserved. This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

This product or the products depicted herein may be protected by one or more U.S. or international patents or pending patents.

Portions of this product may be derived from the UNIX® and Berkeley 4.3 BSD systems, licensed from UNIX System Laboratories, Inc. and the University of California, respectively.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and FAR 52.227-19.

TRADEMARKS
Sun, Sun Microsystems, the Sun Logo, SunSoft, the SunSoft Logo, 386/ix, ARCHITECH, Easy Windows, NFS, Security, and VP/ix are trademarks or registered trademarks of Sun Microsystems, Inc., licensed to SunSoft, Inc., a Sun Microsystems, Inc. business. UNIX is a registered trademark of UNIX System Laboratories, Inc. in the United States and other countries. Adaptec is a registered trademark of Adaptec, Inc. UnTerminal is a trademark of Advanced Micro Research. Apple and Macintosh are registered trademarks of Apple Computer, Inc. Archive is a trademark of Archive Corporation. ICC is a trademark of Bell Technologies. Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may also be a trademark of various telephone companies around the world. Compaq and Compaq 386 are trademarks or registered trademarks of Compaq Computer Corporation. DELL is a trademark of Dell Computer Corporation. DigiCHANNEL MC/Xi is a trademark of DigiBoard, Inc. DEC and VT100 are trademarks or registered trademarks of Digital Equipment Corporation. Hayes, Hayes SMARTMODEM 1200, Hayes SMARTMODEM 2400, Telebit, and TrailBlazer are trademarks or registered trademarks of Hayes Microcomputer Products, Inc. Hercules is a trademark of Hercules Computer Technology, Inc. Informix is a registered trademark of Informix Software, Inc. 386, 387, 486, 80386, and Intel are trademarks or registered trademarks of Intel Corporation.

INTERACTIVE and TEN/PLUS are trademarks or registered trademarks of INTERACTIVE Systems Corporation. AT, IBM, Micro Channel, OS/2, PS/2, and XT are trademarks or registered trademarks of International Business Machines Corporation. AccuTrak and Irwin are registered trademarks of Irwin Magnetic Systems, Inc. CoEdit, CodeWatch, Language Processors, Inc., LPI, LPI-C, and New C are trademarks of Language Processors, Inc. LOGITECH is a trademark of LOGITECH, Inc. Lotus and 1-2-3 are registered trademarks of Lotus Development Corporation. X Window System is a trademark of the Massachusetts Institute of Technology. SMB/ix is a registered trademark of Micro Computer Systems, Inc. Microfield, T8, and V8 are registered trademarks of Microfield Graphics, Inc. Microsoft, MS-DOS, MS-NET, and XENIX are trademarks or registered trademarks of Microsoft Corporation. NCR is a registered trademark of NCR Corporation. Motif, Open Software Foundation, OSF, and OSF/Motif are trademarks of Open Software Foundation, Inc. SCO is a trademark of The Santa Cruz Operation, Inc. SMP and SMP+ are trademarks of SecureWare, Inc. LaserView is a trademark of Sigma Designs. Sony is a trademark of Sony Corporation. SunRiver is a registered trademark of SunRiver Corporation. Video Seven is a trademark of Video Seven, Inc. Looking Glass and Visix are registered trademarks of Visix Software, Inc. XEROX is a registered trademark of XEROX Corporation. X/Open is a trademark of X/Open Company Limited in the United Kingdom and other countries. All other product names mentioned herein are the trademarks of their respective owners.

X Window System™ is a product of the Massachusetts Institute of Technology.

INTERACTIVE™ NFS is derived from System V NFS® developed by Lachman Associates, Inc.

The INTERACTIVE SMB/ix® extension is derived from SMB/ix as developed by Micro Computer Systems, Inc.

The INTERACTIVE Security™ extension is derived from the SMP™ and/or SMP+™ products licensed by SunSoft from SecureWare, Inc.

The term ''network information service (NIS)'' is now used to refer to the service formerly known as Yellow Pages®. The functionality remains the same; only the name has changed. The name Yellow Pages is a registered trademark in the United Kingdom of British Telecommunications plc, and may not be used without permission.

Programs described in this manual are copyrighted and their copyright notices may be found in heralds, by using the UNIX System *what* program, and by reading files whose names start with ''coprisc''.

SunSoft cannot assume responsibility for any consequences resulting from this publication's use. The information contained herein is subject to change. Revisions to this publication or new editions of it may be issued to incorporate such changes.

References to INTERACTIVE Systems Corporation should be replaced by SunSoft, Inc.

INTERACTIVE UNIX Operating System Version 3.0.1 – Update Package –

This *INTERACTIVE UNIX Operating System Update Package* contains new and revised pages for the *INTERACTIVE UNIX Operating System Guide*, Version 3.0.1.

Add the following article to your *INTERACTIVE UNIX Operating System Guide*:

INTERACTIVE UNIX Operating System Version 3.0.1 Release Notes

Replace the following articles in their entirety:

Introduction to the INTERACTIVE UNIX System V/386 Release 3.2 Operating System

Documentation Roadmap

Replace pages in the following article as indicated:

INTERACTIVE UNIX Operating System Maintenance Procedures	Replace pp. 111-120

INTERACTIVE UNIX Operating System Guide

CONTENTS

Introduction to the INTERACTIVE UNIX System V/386 Release 3.2 Operating System

CONTENTS

Introduction to the INTERACTIVE UNIX System V/386 Release 3.2 Operating System

INTRODUCTION

The INTERACTIVE™ UNIX® System V/386 Release 3.2 Operating System, Version 3.0 or later from SunSoft provides users with a powerful and flexible working environment. A friendly user interface and tutorial-style documentation make the system easy to learn and easy to use. The system is delivered with a documentation package that contains all the information you need to install and begin using your system.

Many different extensions are available from SunSoft to increase the productivity of your system, including the VP/ix™ Environment, which allows a user to run MS-DOS® (DOS) and DOS applications under UNIX System V Release 3.2, and the TEN/PLUS® User Interface, which provides an easy-to-use interface to the operating system. Other SunSoft extensions provide users with powerful user interfaces and networking capabilities and give programmers a powerful set of tools for building new applications. The extensions are supported by a complete set of technical reference guides. This document provides an overview of the INTERACTIVE Product Family line, which includes:

INTERACTIVE UNIX OPERATING SYSTEM
: Provides an overview of the INTERACTIVE UNIX Operating System and describes the new UNIX System features and SunSoft enhancements available in it.

SUNSOFT OPTIONAL EXTENSIONS
: Describes the optional programming, networking, and user interface extensions that are available to accompany the INTERACTIVE UNIX Operating System.

INTERACTIVE UNIX OPERATING SYSTEM

The INTERACTIVE UNIX Operating System is an enhanced version of the UNIX Operating System and the foundation of the INTERACTIVE Product Family. It provides a powerful and versatile environment for running and developing applications in a UNIX System environment. The INTERACTIVE UNIX Operating System is a multi-user, multi-tasking system based on Release 3.2 of UNIX System V. It conforms to the accepted standard for UNIX implementations, the System V Interface Definition of UNIX System Laboratories, Inc. (USL), a subsidiary of AT&T, and is fully compatible with other USL-certified UNIX System offerings. In addition, the INTERACTIVE UNIX Operating System has been enhanced and tailored by SunSoft for 386™ based systems.

The UNIX Operating System

The UNIX Operating System was originally developed at AT&T Bell Laboratories. Its multi-user, multi-tasking capability allows many users to share the system facilities and perform multiple tasks concurrently. On a single-user system, such as those provided with most personal computers, only one person at a time can use the computer's files, programs, and other resources.

At the time that the UNIX System was developed, most operating systems could only run on the hardware for which they were written. The UNIX System was revolutionary because it was not dependent on the type of hardware used. This meant that UNIX Systems could be moved to new hardware technologies as they became available, with very few modifications. Today, the UNIX Operating System is available on a wide range of hardware – from small personal computers to the most powerful mainframes – from a multitude of hardware and software vendors.

The UNIX System has been in the commercial market since INTERACTIVE Systems Corporation first offered IS/1. Many versions of the UNIX System have subsequently been released by AT&T (and now, USL), but the most popular version today is known as UNIX System V. The UNIX System has changed through the years, but most of the basic tools and concepts have endured the test of time.

UNIX System Features

UNIX System V Release 3.2 offers enhanced functionality over previous System V releases in a number of areas, including XENIX® System V compatibility, screen management, memory management, and system security. In addition, the UNIX System features Internationalization Support, a STREAMS facility, and shared libraries.

XENIX Binary and Source Code Compatibility permits XENIX System V binary applications to execute on UNIX System V/386 Release 3.2 transparently and without modification. Applications in binary form developed for XENIX System V/386 (Release 2.2.0 and later) and XENIX System V/286 (Release 2.0 and later) will execute without recompiling. Source code for XENIX System V/386 programs, applications, and device drivers can be compiled and linked without modification. In addition, support is now provided for XENIX V/386 system call extensions and several XENIX commands to enhance compatibility.

Internationalization includes support for 8-bit code sets. Commands such as `cat`, `vi`, `grep`, and `ls` now handle code sets where all 8 bits are used. Alternate character classification and conversion rules are also supported. The language and format of the date and time may also be specified. Commands such as `cpio`, `date`, `ls`, and `mount` provide the date and time in the language and format specified.

The STREAMS facility provides a uniform method of implementing network protocols and supporting different network media. It may be implemented over several different protocol families, including TCP/IP (Transmission Control Protocol/Internet Protocol), StarLAN, and OSI. Shared libraries reduce disk and memory requirements by allowing all UNIX System programs to use a single copy of the runtime library.

UNIX System V Release 3.2 Modules

The following UNIX System V Release 3.2 modules (as defined in the System V Interface Definition) are delivered as part of the INTERACTIVE UNIX Operating System:

- Base UNIX System
- Kernel Extension
- Basic Utilities Extension
- Advanced Utilities Extension

- Administered System Extension
- Terminal Interface Extension

All of the components listed above are described in detail in the complete UNIX System V/386 Release 3.2 documentation available from SunSoft. There are also a number of excellent books commercially available that describe the UNIX Operating System and its components. Refer to the ''Documentation Roadmap'' in this guide for information about additional documentation.

Enhancements from SunSoft

The INTERACTIVE UNIX Operating System includes a number of enhancements designed to maximize the speed and performance of UNIX System V Release 3.2, to make it easier to install and use, and to add to its functionality. This section describes a few of the most important enhancements.

- **Improved Installation and System Administration**
  SunSoft has improved and simplified the UNIX System installation procedures for 386-based systems. The user interface to installation has been completely reworked using the Character User Interface (CUI) Toolkit, which provides a full-screen, full-color interface with pull-down menus, forms, on-line context-sensitive help, and a pleasing look-and-feel. As a result, installation requires a far less technical user than in the past.

  SunSoft has used the same interface to rework system administration and achieve the same result: less technical users can now perform all routine system administration tasks, as well as many of the less routine ones, with the same helpful and easy-to-use interface.

  A few of the many tasks facilitated by the new system administration program include installation and removal of software, diskette and fixed disk formatting, backing up the fixed disk, setting and changing system passwords, and setting up the system to use a modem or UUCP.

- **Documentation**
  SunSoft supplies a set of documentation tailored specifically to the INTERACTIVE UNIX Operating System that enhances the traditional set of UNIX System documentation, which is informative but difficult to use. Most documents are shipped in convenient three-ring binders. The manual entries contained in the *INTERACTIVE UNIX System*

V/386 Release 3.2 User's/System Administrator's Reference Manual are delivered as an installable subset, as well as in printed form.

- **Kernel Configuration Link Kit**
  The INTERACTIVE UNIX System Kernel Configuration Link Kit, `kconfig`, provides a system for configuring, linking, and installing a new kernel. It has also been reworked using the CUI Toolkit. This utility simplifies the necessary tasks for the novice user while maintaining functionality desired by experienced users. Menus are provided to save various configurations and install alternate kernels. The INTERACTIVE UNIX System Kernel Configuration Link Kit provides a menu option that allows the addition of both predefined memory size-based tunable parameters and individual tunable parameters to the current configuration. As with installation and system administration, the `kconfig` program uses the same visually pleasing and simple user interface and provides plenty of on-line help.

- **Performance Optimization**
  SunSoft includes two features that significantly increase data throughput:

  — **High Performance Device Driver**
  SunSoft's High Performance Device Driver (HPDD) provides improved disk throughput and allows several drivers to share a hardware controller. The HPDD employs a set of standard routines that implement block I/O drivers and character drivers for a variety of controller hardware. Both disk and cartridge tape drivers are supported. The High Performance Device Driver was developed with the following goals in mind:

    - To increase performance, especially by taking advantage of the SunSoft full capabilities of current peripheral controllers.
    - To support a larger number of controllers.
    - To make it easy to add support for new kinds of controllers, disks, and related magnetic and optical peripherals.

  The HPDD is *more* than just a device driver; it is actually a subsystem in the kernel that was developed to support state-of-the-art peripherals and controllers. It contains a large collection of disk- and tape-oriented subroutines. This central HPDD service package "knows" about many of the capabilities built into today's controllers, such as command chaining, multiple seeks, scatter-gather I/O, SCSI operations, and so on. The hardware-independent core of the HPDD performs all the "strategizing" needed to ensure optimal disk performance for your specific hardware configuration. For example, the HPDD fully supports 1:1 interleave with disk controllers that offer it.

In Version 3.0 or later of the INTERACTIVE Operating System, SunSoft provides controller-specific HPDD disk modules for a variety of ST-506, RLL, ESDI, and SCSI controllers for machines with ISA, EISA, and MCA bus technologies.

— **Fast File System**
The Fast File System (FFS) is a collection of enhancements to UNIX System file system support code in the System V kernel that radically improves disk file I/O performance. It supports the standard, unmodified UNIX System file system and provides sequential file I/O data rates well above those of the standard kernel. The FFS is of particular value to the INTERACTIVE UNIX Operating System because the performance of the UNIX System on personal computers is usually limited by disk throughput, not the computational capacity of the processor.

The much higher sequential file throughput is the result of two basic block-handling techniques:

- Files are allocated from a free-block bit map in clusters of physically contiguous disk blocks. This map is built by reading the entire free list when a file system is mounted.
- When a file is processed sequentially, the physical disk read or write operations are performed on clusters of contiguous blocks, rather than one block at a time. By assuring that data for sequentially-accessed files is in memory before it is asked for by a program, the number of disk accesses required can be greatly reduced.

Reading and writing contiguous block clusters increases disk I/O throughput because it requires fewer head seek operations and less physical interleaving of disk sectors. Use of these methods contrasts with other enhanced file systems, such as Berkeley, which gain speed by completely restructuring the file system, thus making them incompatible with the standard System V/386 file system.

The Fast File System works only with a standard System V 1K-block file system on fixed disks and diskettes. Specifically, it does not work with XENIX file systems, with the USL 2K-block file system, or with mounted DOS file systems. These exhibit previous performance parameters. Our measurements indicate that file system throughput is likely to be much better with a 1K-block file system and the FFS than with the 2K-block file system for most applications.

— **Enhanced File System Layout**
The file system layout is designed to contain all UNIX System-specific structures in the UNIX System partition. This feature allows the user to retain existing DOS partitions when installing the INTERACTIVE UNIX Operating System.

— **Utilities to Access DOS Files**
SunSoft has developed two facilities to assist INTERACTIVE UNIX System users in accessing DOS files from the UNIX System:

  — **Dossette File Exchange Utility**
  This facility provides the ability to manipulate DOS-format file systems under the UNIX System and to move files between DOS and UNIX System file systems. It works with both DOS diskettes and fixed disk partitions.

  — **Integrated DOS File System Support**
  This facility also allows users to access DOS file systems on diskettes or fixed disk partitions while running the UNIX System. With the INTERACTIVE UNIX System Integrated DOS File System Support feature, mounted DOS file systems appear to be ordinary UNIX System file systems to the user. This facility greatly simplifies the interface between the UNIX System and DOS; the DOS file system is completely accessible to multiple UNIX System or DOS-under-VP/ix processes. Files and processes are protected with the UNIX System file and record locking functions.

— **Internationalization**
The INTERACTIVE UNIX System contains a number of proprietary utilities and improvements with respect to internationalization. The `ttymap` utility can be used to remap characters to properly support different keyboards and terminals. It supports any 8-bit code set and features character mapping on input and output, support for deadkeys and compose character sequences, and a toggle key to temporarily disable the mapping

from within an application. When invoked from the console, it can also be used to change the translation of scancodes. The required mapping can be specified in a text file. Mapfiles for all major European keyboards are supplied with the system.

The `getty` and `stty` utilities are enhanced so that mapping can be activated even before a user logs in. `loadfont` is a utility that makes it possible to change the font that is used on the console, if the video card supports it. Fonts for the IBM® 437 and 850 codepage, as well as the ISO8859-1 standard, are supported. If needed, user-defined fonts can be specified in a text file that is then read by the `loadfont` program. See *getty*(1M), *loadfont*(1), and *ttymap*(1) for details. See also *loadfont*(5) in the *INTERACTIVE Software Development System Guide and Programmer's Reference Manual.*

— **Device Drivers**
To provide a versatile and highly configurable product, SunSoft has assembled a set of device drivers to support standard peripheral boards including the following:

— Standard keyboards (92 and 101 keys)

— Display adapters (monochrome, CGA, EGA, VGA, and Hercules ™

— Standard serial ports (COM1 and COM2)

— Diskettes (5.25- and 3.5-inch media)

— Fixed disks (RLL, SCSI, and ESDI)

— Parallel printers

— Streaming cartridge tapes (both directly attached and SCSI)

SunSoft's Kernel Configuration Link Kit allows users to easily configure new device drivers.

— **Virtual Terminal Support**
The Virtual Terminal Support facility enables users to run multiple, full-screen applications on the system console. Applications may be run concurrently by using a "hot-key" to switch between active processes. This facility supports up to eight concurrent virtual terminal sessions on the system console and supports graphics applications.

— **Berkeley Utilities**
SunSoft includes two important Berkeley utilities with the INTERACTIVE UNIX Operating System:

  — **Sendmail**
  `sendmail` is a general purpose internetwork mail routing facility. It can initiate and receive mail transfers between local users over UNIX System-to-UNIX System copy connections (UUCP) as well as over Transmission Control Protocol/Internet Protocol (TCP/IP) connections.

  — **C Shell**
  The C Shell is the standard shell used in the Berkeley UNIX System. Its many popular features include a powerful command language syntax, which resembles the C programming language; a history mechanism that allows the user to re-execute previous command lines without having to retype them; and an aliasing mechanism that allows the user to define a set of command macros.

— **POSIX and XPG3 Compliance**
The INTERACTIVE UNIX Operating System conforms to a number of operating systems' standards:

- The traditional SVID standard conformance adopted by System V.
- The POSIX/FIPS standard as specified by *IEEE Std. 1003.1-1988* and FIPS 151-1.
- The XPG3 standard as specified by the *X/Open Portability Guide*, Issue 3. This standard is built on POSIX 1003.1 and contains extensions useful for internationalizing applications.

The traditional SVID environment and the POSIX/FIPS/XPG3 environment are supported concurrently. The choice between environments is made at compilation time; only programs compiled for POSIX/FIPS/XPG3 will execute with that environment. For information about creating POSIX/FIPS-compliant applications, refer to the "INTERACTIVE Software Development System Release Notes." For information about XPG3 conformance and internationalized environments, refer to the *International Supplement Guide* (not included in domestic product shipments).

SUNSOFT OPTIONAL EXTENSIONS

SunSoft provides a variety of extensions designed to further enhance the power of your 386-based system. These extensions provide UNIX System development tools, advanced networking utilities, and powerful user interface capabilities.

VP/ix Environment

The VP/ix Environment allows a user to run DOS and DOS applications under UNIX System V Release 3.2. With the VP/ix Environment, multiple users can run multiple UNIX System and DOS applications simultaneously. For example, you could work on a Lotus® 1-2-3® spreadsheet and access an Informix® database at the same time. In addition, the VP/ix Environment provides a completely integrated file system that is accessible from both DOS and the UNIX System, making it unnecessary to separate your DOS and UNIX System files and applications. A copy of MS-DOS, licensed for use in a multi-user environment, is provided with the VP/ix Environment to enhance performance.

TEN/PLUS Environment

- *TEN/PLUS User Interface*
  The TEN/PLUS User Interface is designed to make the system easy to learn and easy to use. It contains a very powerful editor with consistent functions. The same command will have the same result, regardless of whether you are reading a mail message, writing a computer program, or updating your calendar. It is not necessary to learn a new set of commands each time you use a new application or try a new task. Release 2.2.5 of the TEN/PLUS User Interface features variable size window displays and improved menu instructions. The TEN/PLUS Environment is not sold as a separate item but is included with all bundled packages.

- *TEN/PLUS Mail System*
  The TEN/PLUS Environment includes the TEN/PLUS Mail System, an electronic message system that provides each user with a private mailbox. It allows the user to send messages to one or more users or mailing lists, with copies or blind copies; reply to, forward, print, delete, or restore messages; move or copy messages; execute other programs without leaving the mail system; review past correspondence in the primary and secondary mailboxes; and send mail to and receive mail from remote systems. Its TEN/PLUS interface provides

these features through user-friendly forms with menus, function keys, and on-line help. It interfaces with Berkeley `sendmail`, USL's `mail`, and other mail systems.

INTERACTIVE Software Development System

- *INTERACTIVE Software Development System*
  The INTERACTIVE Software Development System is available for programmers who plan to use the INTERACTIVE UNIX Operating System as a software development environment. It provides the UNIX System V Release 3.2 facilities for software development. This extension includes the C compiler, the Source Code Control System (SCCS), `make`, `yacc`, `lex`, and other development tools, such as the Extended Terminal Interface, all based on the USL Software Generation System. The manual entries contained in the *INTERACTIVE Software Development System Guide and Programmer's Reference Manual* are delivered as an installable subset, as well as in printed form. In addition, a POSIX and XPG3 development environment is provided, containing an enhanced `cc` utility and a library of POSIX and XPG3 functions.

- *Software Integration Tools*
  This subset contains easy-to-use tools that take application software and build a subset that can be installed with the `sysadm` facility.

- *New C*
  New C™ is Language Processors, Inc.'s (LPI™) powerful implementation of ANSI C, which incorporates an extensive array of features, functionality, and pre-processor improvements. ANSI C provides the user with the ability to create more reliable, maintainable, and highly portable code. Execution is greatly improved, and New C saves the developer time through complete sentence error messages that pinpoint the location of the error. New C is ANSI-compliant and can be used with the POSIX/XPG3 library to develop applications that conform to *IEEE Std. 1003.1–1988.*

- *CodeWatch*
  This version of CodeWatch™ (developed by LPI) is a powerful, interactive source-level debugger that operates exclusively with New C to provide excellent source-level debugging capabilities for the INTERACTIVE Software Development System. CodeWatch works on actual New C source code without using an intermediate language, which results in highly accurate debugging of the source code. With

CodeWatch, the program variables can be tracked and the conventions and symbols of the source language can be used, which eliminates the need to learn the machine language of the computer.

- *CoEdit*
  CoEdit™ (developed by LPI) is a customizable, feature-rich, language-sensitive editor. It performs syntax checking, expression evaluation, configuration from within the editor, and automatic error location. It includes context-sensitive help and many productivity-enhancing and convenient programming features, as well as a macro programming environment, complete with a compiler and debugger for macros. Enhanced block, locate and replace, and tab functions are included in this powerful editor, which supports New C.

INTERACTIVE TCP/IP

INTERACTIVE TCP/IP facilities provide the standard TCP/IP data transfer services, such as internetwork routing between network interfaces, security facilities, control of processing to minimize transmission, and three classes of internetwork addresses. TCP/IP supports two programming interfaces, the USL Transport Layer Interface (TLI) and the Berkeley Software Distribution (BSD) socket interface, which provide access to the data transfer services of the underlying network protocols. Additional applications, such as `telnet` and `ftp`, are also provided, along with a substantial number of networking commands. The software includes device driver components for each supported Ethernet board. INTERACTIVE TCP/IP provides support for user-level applications and network services such as the INTERACTIVE NFS® Extension to enhance network performance in diverse operating environments.

INTERACTIVE NFS Extension

INTERACTIVE NFS, derived from System V NFS developed by Lachman Associates, Inc., provides support for Sun Microsystems® NFS Release 3.4. It provides network services that allow users in heterogeneous computing environments to share files, access remote resources, and mount file systems across a network. The NFS distributed computing file system is a de facto industry standard for transparent file access among differing hardware architectures and operating systems, and includes network administration facilities to control the flow of messages over the network. Features include file and record locking, program execution on remote systems, and remote, heterogeneous communication.

INTERACTIVE X11 Runtime System

The INTERACTIVE X11 Runtime System contains all the necessary software for executing X applications on the INTERACTIVE UNIX Operating System. This includes the shared Runtime Xlib library, servers that support a large variety of displays and input devices, and the Installation Program. Among the clients included are a terminal emulator, a window manager, calculator programs, and clocks.

The INTERACTIVE X11 Runtime System also contains the Open Software Foundation™ (OSF™) Motif™ Window Manager (MWM), which provides the standard graphical user interface that provides a Presentation Manager "look-and-feel." MWM allows users to manage their application windows via such actions as moving, resizing, reducing to icons, restoring, and many others. MWM allows both color and black and white windows and offers INTERACTIVE X11 users a common user interface that is available across a wide range of UNIX System platforms.

The INTERACTIVE X11 Runtime System also includes the Easy Windows™ Environment.

INTERACTIVE X11 Development System

The INTERACTIVE X11 Development System includes libraries, documentation, and the necessary software for creating INTERACTIVE X11 applications. The X Toolkit library provides tools for simplifying the design of application user interfaces under INTERACTIVE X11.

INTERACTIVE Motif Development System

The INTERACTIVE Motif Development System, derived from OSF/Motif, Revision 1.1.1, licensed by SunSoft from the Open Software Foundation, Inc., includes all the tools and information needed to develop applications that are compliant with the OSF/Motif Style Guide. The Style Guide is a document that specifies the constraints and guidelines to follow when designing applications. The Motif Toolkit includes a rich collection of gadgets and widgets for building applications, and the User Interface Language allows developers to create simple text files that describe visual properties.

INTERACTIVE Looking Glass Professional Desktop Manager

The INTERACTIVE Looking Glass Professional™ Desktop Manager, derived from the product by Visix® Software, Inc., is an X-based desktop manager that provides a user-friendly and easy-to-use icon-based and mouse-driven interface. Implemented in compliance with the OSF/Motif* "look-and-feel" standard, the INTERACTIVE Looking Glass Professional Desktop Manager allows users to view files and directories in both text and icon forms, navigate the UNIX System quickly and easily, and launch applications. The INTERACTIVE Looking Glass Professional Desktop Manager provides the final layer of system software that combines the power of the INTERACTIVE UNIX Operating System with the operating ease and friendliness of an Apple® Macintosh® computer. The INTERACTIVE Looking Glass Professional Desktop Manager is not sold as a separate item, but is part of two solution bundles: the INTERACTIVE Workstation Platform and Workstation Developer.

INTERACTIVE SMB/ix Extension

The INTERACTIVE SMB/ix® Extension, derived from SMB/ix as developed by Micro Computer Systems, Inc., allows a DOS-based client to access files and printer services on an INTERACTIVE UNIX System server. Based on the Server Message Block (SMB) protocol, INTERACTIVE SMB/ix provides compatibility for IBM PC LAN or MS-NET™ networking applications.

The INTERACTIVE SMB/ix Extension consists of two components, the INTERACTIVE SMB/ix Server module, which runs on the INTERACTIVE UNIX System host, and the INTERACTIVE SMB/ix UNIX System Client Support module, which runs on the PC (and is provided free of charge). INTERACTIVE SMB/ix integrates DOS and UNIX System file systems into a single file system, operates as a process on the INTERACTIVE UNIX Operating System concurrently with other UNIX System processes, and includes a menu-driven system administration program.

The INTERACTIVE SMB/ix Extension increases the power and value of existing PCs and preserves investment in hardware, applications, and training. It can also be used to make UNIX System peripherals available to more users.

INTERACTIVE Security Extension

The INTERACTIVE Security™ Extension, derived from the SMP™ and/or SMP+™ products licensed by SunSoft from SecureWare, Inc., is an optional extension to the INTERACTIVE UNIX Operating System that raises its commands and utilities to the C2 class of trust as defined by the *Trusted Computer System Evaluation Criteria* (known as the *Orange Book*). The INTERACTIV Security Extension is a usable and easily administered system that goes beyond the requirements set forth in the *Orange Book*. A collection of programs called the Trusted Computing Base (TCB) maintains the parts of the system's state that are related to security. The TCB consists of a modified kernel and the trusted utilities that reference and maintain relevant security data. The INTERACTIVE Security Extension corrects, modifies, and adds to the standard utilities and operating system to define its TCB. The TCB implements the security policy of the system. Much of the software with which the system administrator interacts is part of the system's TCB. The INTERACTIVE Security Extension provides a menu-driven, administrative interface to help maintain the TCB.

HOW TO USE THIS GUIDE

This guide contains the basic documentation you need to install, maintain, and use the INTERACTIVE UNIX Operating System, Version 3.0 or later. Whether you are an experienced programmer or a novice user, be sure to read the next few pages of this document. They will tell you what is contained in this guide and in the *INTERACTIVE UNIX System V/386 Release 3.2 Guide for New Users*, and how to use this information to your best advantage.

What's Included

The *INTERACTIVE UNIX System V/386 Release 3.2 Operating System Guide* includes:

- **INTERACTIVE UNIX Operating System Release Notes**
  Provides information about software enhancements, bug fixes, and special requirements for the current release of the INTERACTIVE UNIX Operating System.
- **INTERACTIVE UNIX Operating System Installation Instructions**
  Provides step-by-step instructions for installing, initializing, and logging in to the INTERACTIVE UNIX Operating System.
- **INTERACTIVE UNIX Operating System Maintenance Procedures**
  Provides a detailed overview of the advanced procedures required to keep the INTERACTIVE UNIX Operating System running smoothly.
- **An Introduction to Sendmail**
  Provides information of use to `sendmail` users, including an overview of `sendmail`, a list of its configuration files, and explanations of how to check your mail system and to forward mail.
- **Sendmail – An Internetwork Mail Router**
  Describes the design goals for `sendmail` and gives an overview of its basic functions. It also discusses usage details, compares `sendmail` to other internet mail routers, and evaluates `sendmail`, including future plans.
- **Sendmail Installation Instructions**
  Provides installation instructions specific to using `sendmail` on the INTERACTIVE UNIX Operating System.
- **Sendmail Installation and Operation Guide**
  Provides the information needed to do a basic `sendmail` installation and the day-to-day information needed to maintain the mail

system. It also describes some parameters that may be changed, the command line arguments, and information about the configuration file. The appendixes give a brief but detailed explanation of a number of features not described in the rest of the document.

- **An Introduction to the C Shell**
  Provides a basic explanation of the C shell, including its capabilities and features, and a glossary that lists the special characters, terms, and commands used.
- **POSIX Compliance Document**
  This document is required by the Institute of Electrical and Electronic Engineers (IEEE) for an implementation claiming conformance to *IEEE Std. 1003.1-1988* (section 2.2.1.2). It has the same structure as the standard, with the information presented in the appropriately numbered sections.
- **Documentation Roadmap**
  Outlines the complete documentation set available from SunSoft. It describes the documentation that accompanies the INTERACTIVE UNIX Operating System and additional, supplemental documentation available to you.
- **Reader's Comment Form**
  Provides you with a way to tell us what you like or dislike about this guide and to send us your ideas for making it even better.
- **International Supplement Guide**
  Provides information on installation, features, and internationalized applications development.

The *INTERACTIVE UNIX System Guide for New Users* includes:

- **INTERACTIVE UNIX Operating System Primer**
  Provides a tutorial introduction to the basic components and features of the INTERACTIVE UNIX Operating System and describes basic UNIX System concepts.
- **System Administration for New Users of the INTERACTIVE UNIX Operating System**
  Provides a description of the basic system administration procedures needed to manage the INTERACTIVE UNIX Operating System and includes tutorial information for performing basic system administration tasks where appropriate.

Where to Begin

The following outline provides some suggested ways to use this guide:

- **If you are a beginner . . .**
  Read this introduction to get to know what the INTERACTIVE UNIX Operating System is all about. Next, work through the ''INTERACTIVE UNIX Operating System Primer'' in the *INTERACTIVE UNIX System Guide for New Users* to learn the basics of using the UNIX System. Then refer to ''System Administration for New Users of the INTERACTIVE UNIX Operating System'' in that same document to learn the basic procedures needed to manage your system.

- **If you are an experienced UNIX System user . . .**
  Read this introduction, which outlines the enhancements and features included in Version 3.0 or later of the INTERACTIVE UNIX Operating System. Then, read the ''INTERACTIVE UNIX Operating System Release Notes'' for information on the latest enhancements to the INTERACTIVE UNIX Operating System. For more detailed and technical information about the UNIX System, refer to the documentation listed in the ''Documentation Roadmap.''

- **If you are installing the system . . .**
  Read and follow the steps outlined in the ''INTERACTIVE UNIX Operating System Installation Instructions.'' Once you have completed the basic system installation, go to ''System Administration for New Users of the INTERACTIVE UNIX Operating System'' in the *INTERACTIVE UNIX System Guide for New Users* if you need basic system administration help, or to the ''INTERACTIVE UNIX Operating System Maintenance Procedures'' for information about how to perform more advanced system maintenance tasks and tailor the system to match your requirements. If you need further information, refer to the documents described in the ''Documentation Roadmap.''

- **If you want the latest system information . . .**
  Read this introduction for a description of the enhancements and features included in Version 3.0 or later of the INTERACTIVE UNIX Operating System. Then, read the ''INTERACTIVE UNIX Operating System Release Notes'' for up-to-the-minute information on what's new in the INTERACTIVE UNIX Operating System.

- **If you want supplemental documentation . . .**
  Refer to the documentation listed in the ''Documentation Roadmap'' for more detailed and technical information on the UNIX System.

CONVENTIONS USED

Throughout this guide and other INTERACTIVE Product Family documentation, boxed words indicate keys on your keyboard. For example, $\boxed{\text{ENTER}}$ refers to the key that moves the cursor to the next line. When you are instructed to type a command, the command must always be followed by using the $\boxed{\text{ENTER}}$ key.

⇒ Keys on your keyboard may be labeled differently than those shown in the documentation. For example, the $\boxed{\text{ENTER}}$ key is labeled $\boxed{\text{RETURN}}$ on some systems. If your hardware or software vendor supplies additional documentation with your system, read that documentation for information on key names before you continue.

Illustrations of computer screen displays, file names, directory names, and commands are printed in a typeface called `constant width`. Constant width text looks like the text produced by most typewriters. Whenever you are instructed to type anything shown in constant width in this guide, type it exactly as it is shown.

Italics indicate the variables in a command or instruction format. In actual use, a real name or number replaces the italicized text. For example, the sequence `rm` *filename* shows the format for removing a file. The word *filename* is replaced with the name of a real file that you would like to remove from your system. Italics are also used for emphasis and when new terminology is introduced. New terms in italics can be found in the glossary at the end of the ''INTERACTIVE UNIX Operating System Primer'' and at the end of other user-level documents.

References of the form *name*(*n*) refer to an entry called *name* in section *n* of the reference manual or manual entries associated with that product or as stated in the documentation.

In the ''INTERACTIVE UNIX Operating System Primer'' and other tutorial documents, new commands are introduced in a double-boxed table. This display provides basic information about the command's format (usage), description, options, and arguments.

DOCUMENTATION REFERENCES

Throughout this guide and the INTERACTIVE Product Family documentation, the following full documentation titles will be referenced in shortened versions as follows:

Full Title	*Shortened Version*
INTERACTIVE UNIX System V/386 Release 3.2 Operating System Guide	INTERACTIVE UNIX Operating System Guide
INTERACTIVE UNIX System V/386 Release 3.2 Guide for New Users	INTERACTIVE UNIX System Guide for New Users
INTERACTIVE UNIX System V/386 Release 3.2 User's/System Administrator's Reference Manual	INTERACTIVE UNIX System User's/System Administrator's Reference Manual
INTERACTIVE UNIX System V/386 Release 3.2 User's Guide	User's Guide
INTERACTIVE Software Development System Guide and Programmer's Reference Manual	INTERACTIVE SDS Guide and Programmer's Reference Manual

Note that in documentation prepared by AT&T, references to certain AT&T books now refer to information present in different INTERACTIVE documents.

AT&T Document	*INTERACTIVE Document*
UNIX System V/386 Release 3.2 Operations/System Administration Guide	INTERACTIVE UNIX Operating System Maintenance Procedures
	System Administration for New Users of the INTERACTIVE UNIX Operating System

UNIX System V/386 Release 3.2 Release Notes	INTERACTIVE UNIX Operating System Release Notes
UNIX System V/386 Release 3.2 User's/System Administrator's Reference Manual	INTERACTIVE UNIX System V/386 Release 3.2 User's/System Administrator's Reference Manual
UNIX System V/386 Release 3.2 Programmer's Reference Manual	INTERACTIVE Software Development System Guide and Programmer's Reference Manual

FOR MORE INFORMATION

The documentation included in this guide provides information about how to install, use, and maintain the INTERACTIVE UNIX Operating System and includes information about the UNIX System. This guide does not include complete technical information on all aspects of the UNIX System. For a more complete listing of related documentation, refer to the "Documentation Roadmap" provided in this guide.

INTERACTIVE UNIX Operating System
Version 3.0.1
Release Notes

CONTENTS

INTERACTIVE UNIX Operating System Version 3.0.1 Release Notes August 1992

1. INTRODUCTION

INTERACTIVE™ UNIX® System V/386 Release 3.2, Version 3.0.1 from SunSoft incorporates additional hardware support, enhanced tape capability, e.g., read and write variable blocksize, and a number of bug fixes. These release notes supplement the Version 3.0 release notes and describe only the differences between Version 3.0 and Version 3.0.1.

2. ADDITIONAL HARDWARE SUPPORT

Important features added in Version 3.0.1 include the following:

- Support for removable media technology – the INTERACTIVE UNIX System now supports Bernoulli and Syquest mass storage devices. These devices combine the performance and capacity of traditional fixed disks with the advantages of transportability and expandability. This support does not extend to installation on removable media.
- Olivetti EISA SCSI controller, ESC-X host adapter
- AMI Fast SCSI controller, running in Adaptec® 154x compatible mode, or BusTek 742 enhanced mode
- Cornerstone PC202A Graphics controller

Also supported in this release is Intel's LP486 series line of EISA machines, using:

- NCR® 53C700 SCSI I/O Processor
- Embedded VGA controller
- Embedded IDE controller
- Intel® 82596 embedded Ethernet controller

3. SOFTWARE CORRECTIONS AND FIXES

Version 3.0.1 fixes over 100 known bugs; some significant ones include:

- The DPT SCSI driver now flushes the controller cache when `init 6` is executed, preventing a condition which could occasionally cause some corruption of the `root` file system.
- The interrupt-vector conflict for the Adaptec 1742 controller has been fixed.
- The AHA1540 controller can now be installed as a secondary controller after configuring a primary MCA controller.
- E-mail no longer generates multiple copies of messages to recipients.
- Tape reads now succeed on tapes written on various devices, with varying blocking factors.
- It is now possible, using `sysadm backup`, to back up the `root` file system, excluding `/usr`.
- A `vi` ''wrap margin'' bug that corrupted lines where the original line length was greater than the new margin setting has been fixed.
- In systems that create a separate mountable `/usr/spool` file system, both `/usr` and `/usr/spool` are now properly mounted.
- E-mail has been enhanced to increase the buffer size of the `To` field from 4096 bytes to 12288 bytes, allowing longer lists of recipients for individual mail messages.

4. CAVEATS, PROBLEMS, AND WORKAROUNDS

The following deficiencies are known to exist in Version 3.0.1. Where available, workarounds are presented.

- The KornShell does not support job control.
- The KornShell does not support filename globbing within a DOS partition, e.g.:

  ```
  ls *
  ```

  returns

  ```
  *
  ```

  rather than a list of file names.

- The `NCR700` driver for the on-board NCR53C700 SCSI I/O Processor on the Intel LP486 series machines does not support multi-threaded I/O or multiple LUNs.
- Exabyte/Sun 8mm tape cannot automatically determine blocksize. Users must explicitly specify:

  ```
  mt -b 1
  ```

  for 512 bytes/block, or

  ```
  mt -b 2
  ```

  for 1024 bytes/block when using the `cpio`, `dd`, and `tar` commands.

5. INSTALLATION

This section applies only to those users who have received a Platform Update to update their existing Version 3.0. Anyone who has received a full Version 3.0.1 release should refer to ''INSTALLING VERSION 3.0'' in the ''INTERACTIVE UNIX Operating System Version 3.0 Release Notes'' and to the ''INTERACTIVE UNIX Operating System Installation Instructions'' for instructions on installing their system.

The Version 3.0.1 Platform Update is distributed on the following diskettes which represent full replacements for those INTERACTIVE UNIX Operating System subsets that have changed between versions 3.0 and 3.0.1, as well as for any update disks:

- Boot
- Install
- 4 Core diskettes
- 1 diskette containing the Basic Networking and High Sierra File System subsets
- 3 diskettes containing the File Management and Kernel Configuration subsets
- 1 diskette containing the User's Manual Entries
- 2 diskettes containing the International Supplement
- INTERACTIVE Network Drivers diskettes – these will update the network drivers for those who have INTERACTIVE TCP/IP.
- INTERACTIVE X11 Update diskettes – these will update INTERACTIVE X11.

To install the update, replace all the above 3.0 subsets with their 3.0.1 counterparts, and perform an update as described in section 4 of the "INTERACTIVE UNIX Operating System Installation Instructions." Be sure to update all the subsets for which you had a 3.0 version installed.

1. Install your INTERACTIVE UNIX Operating System according to the instructions referred to above, **but use the new (Version 3.0.1)** ***Boot*** **and** ***Install*** **diskettes as replacements for the Version 3.0** ***Boot*** **and** ***Install*** **diskettes.**
2. Choose the `update` option and load the *Core* diskettes as instructed.
3. Use `sysadm installpkg` to install the other software.
4. Use `kconfig` to rebuild and reinstall your kernel. Shut down and reboot your system. The update to Version 3.0.1 is now complete.

6. NEW FEATURES

6.1 KornShell

The KornShell (see *ksh*(1)) is a UNIX System command interpreter that behaves like the Bourne Shell, has some C shell features, and has new additional features of its own. One key feature is the ability to repeat previous commands, either as they were originally entered, or to edit them before execution. The KornShell has become the command interpreter of choice for most programmers and is now supplied as part of the INTERACTIVE UNIX Operating System.

6.2 International Supplement

The International Supplement is now included in all shipments of the INTERACTIVE UNIX System. This package contains internationalized versions of the most popular UNIX System utilities, such as `date`, `sort`, and `ls`. It also contains data files to support keyboards, date and time formats, and fonts for most European languages and countries.

7. DOCUMENTATION NOTES

The following products referred to in parts of the documentation are no longer available from SunSoft:

INTERACTIVE Network Connection Facilities

INTERACTIVE Ported NetWare

INTERACTIVE Text Processing Workbench

INTERACTIVE UNIX Operating System Version 3.0 Release Notes

CONTENTS

INTERACTIVE UNIX* Operating System Version 3.0 Release Notes October 1991

1. INTRODUCTION

INTERACTIVE UNIX System V/386 Release 3.2, Version 3.0 incorporates numerous new features, enhancements, and bug fixes into INTERACTIVE's enhanced UNIX System V/386 Release 3.2 for Intel*-based 386* and 486* systems. These release notes contain the following information:

NEW FEATURES IN VERSION 3.0
: This section describes all of the new features and enhancements to existing features made since Version 2.2.1 of the INTERACTIVE UNIX Operating System, which was the direct predecessor of Version 3.0.

SOFTWARE CORRECTIONS AND FIXES
: This section describes all of the bugs that have been fixed since Version 2.2.1 of the INTERACTIVE UNIX Operating System.

HARDWARE NOTES
: This section describes some known idiosyncrasies and incompatibilities of certain hardware supported by Version 3.0, which could not be worked around in the software.

CAVEATS, PROBLEMS, AND WORKAROUNDS
: This section lists certain caveats the user should be aware of and provides a list of bugs known to exist in the release (with workarounds where available).

INSTALLING VERSION 3.0
: This section gives a short overview of the installation and update procedures for Version 3.0. To actually install or update your system, refer to the "INTERACTIVE UNIX Operating System Installation Instructions" in this guide.

Appendix A: SUMMARY OF INTERACTIVE SUBSETS
This appendix lists all of the INTERACTIVE UNIX System subsets with a brief description.

Appendix B: VERSION 2.2.1 INFORMATION
This appendix provides information contained in the release notes for Version 2.2.1 of the INTERACTIVE UNIX Operating System. It is provided for the benefit of those readers who are updating directly from Version 2.2 and may not have access to the Version 2.2.1 release notes. The difference between Version 2.2 and Version 3.0 can be viewed as the sum of the information contained in the main body of the Version 3.0 release notes, which describes the changes between Version 2.2.1 and Version 3.0, and the information contained in this appendix, which describes the changes between Version 2.2 and Version 2.2.1.

2. NEW FEATURES IN VERSION 3.0

The features described in this section have been added to the INTERACTIVE UNIX Operating System since Version 2.2.1.

2.1 Enhanced System Administration

The system administration utility (`sysadm`) has been completely rewritten to take advantage of the full-color INTERACTIVE Character User Interface (CUI) Toolkit that was first introduced in the system installation portion of Version 2.2. In Version 3.0, most of the common system administration functions are performed using pull-down menus, pop-up dialog boxes, and visually pleasing screen forms for data entry. Context-sensitive on-line help is available at all stages of system administration, with the result that most day-to-day administration can be done without a manual.

More than just the user interface of this utility has been rewritten – each of the system administration functions was reviewed and enhanced where necessary to provide more functionality and greater ease of use. In some cases, entirely new areas of functionality were added.

For more information on using the new system, refer to sections 2 and 3 of the "INTERACTIVE UNIX Operating System Maintenance Procedures" in this guide, or simply type the command `sysdemo` or log in using the `sysdemo` login and use the on-line help to navigate through the system and learn how to use it without worrying about harming your system; the `sysdemo` mode allows you to navigate the system, use all the features and fill in any form, without actually executing your requests.

2.2 Enhanced Kernel Configuration

The Kernel Configuration utility (`kconfig`) has also been rewritten to use the INTERACTIVE CUI Toolkit, and as a result the new Version 3.0 `kconfig` is far easier to use and understand. It has also been made modular and extendable; driver and help information that was previously hard-coded into the `kconfig` program is now located in user-editable ASCII files. These files are both easy to modify and easy to translate into other languages.

As with the new `sysadm`, more than just the user interface has been improved; functionality has been substantially enhanced. One example of this is that manual editing of kernel configuration files is no longer required in order to configure new devices at nonstandard addresses and interrupt lines. Another is the ability to easily configure the High Performance Device Driver (HPDD) (referred to in earlier releases as the High Performance *Disk* Driver) to support complex SCSI system configurations.

For more information about how to use the new `kconfig`, refer to sections 7 and 8 of the "INTERACTIVE UNIX Operating System Maintenance Procedures."

2.3 XPG3 Conformance

Standards conformance continues to be a major area of focus for the INTERACTIVE UNIX Operating System. With Version 3.0, strict conformance to the current version of the *X/Open Portability Guide* (XPG3) has been added to the existing compliance with both POSIX 1003.1 and SVID Issue 2. In combination with Version 3.0 of the INTERACTIVE Software Development System and the *International Supplement Guide*, this supports the development and testing of applications with the following features:

- Localized collation sequences
- Internationalized message files
- Localized regular expression handling
- Localized date and monetary format

As with POSIX 1003.1 conformance, XPG3 conformance occurs on a per-process basis. Applications and commands compiled and linked for a POSIX and XPG3 environment adhere to those standards when run, while all others adhere to SVID Issue 2.

2.4 International Supplement

Nondomestic shipments of the INTERACTIVE UNIX Operating System include a new subset called the International Supplement, which contains:

- All of the commands (over 30) specified by XPG3 as having internationalized behavior, such as `awk`, `grep`, `sed`, and so on. These commands adhere to local collation conventions, regular expression handling, date and monetary formats, and localized message files, if supplied. A `locale` (**local**ization environment) must be provided in order to benefit from these features.

- A full sample `locale` for the French language, which may be used as a template for the development of other `locales`. Other `locales` may be supplied by your local reseller.
- Subsets of contributed data files, such as keyboard map files and partial `locales` for various languages. These are provided on an "as-is" basis and are not supported.
- A guide describing these features, as well as others already present in the INTERACTIVE UNIX System, that provides comprehensive information for developers, system administrators, and end users in non-English speaking countries.

It is anticipated that developers in the United States who wish to write applications that conform to XPG3 may require the International Supplement, which is available domestically as a separately purchasable product.

2.5 EISA Support

While earlier releases of the INTERACTIVE UNIX System could be used on machines with EISA bus architectures, the system did not take advantage of any of the powerful features inherent in that architecture, nor did it support EISA disk controllers running in their enhanced EISA mode. Version 3.0 provides full support for this powerful new class of machines, including:

- Support for physical memory over 16 MB on EISA machines. The maximum amount of physical memory supported on these machines is 256 MB.
- Support for a number of EISA disk controllers and SCSI adapters, as described in section 2.6, "Peripheral Support." An EISA disk controller must be used to take advantage of more than 16 MB of memory. When an EISA disk controller is present, DMA to memory over 16 MB is done directly rather than being remapped through memory below 16 MB, which would degrade performance.
- A kernel driver for developers of EISA controllers to use as an interface to the EISA NVRAM. (For more information about EISA support, refer to *eisarom*(7) in the *INTERACTIVE UNIX System User's/System Administrator's Reference Manual*.)

2.6 Peripheral Support

Support has been added in Version 3.0 for the following devices:

- The Adaptec* 1520 SCSI host board adapter
- The Adaptec 1740 EISA SCSI host board adapter
- The BusTek 742 EISA SCSI host board adapter
- The COMPAQ* Intelligent Drive Array EISA disk controller
- The COMPAQ 320/525 MB SCSI tape adapter
- The DPT 2011 ISA SCSI host board adapter
- The DPT 2012 EISA SCSI host board adapter
- The IBM* MCA SCSI host board adapter (for PS/2* Model 95 machines)
- The Future Domain TMC-1680 SCSI host board adapter
- The Future Domain MCS-700 MCA SCSI host board adapter
- The Western Digital 7000ASC SCSI host board adapter
- The Western Digital 7000EX EISA SCSI host board adapter
- All SCSI-compatible CD-ROM drives attached to any supported SCSI host board adapter

The Adaptec 1540 series of SCSI host board adapters and the Future Domain TMC-800 series of SCSI host board adapters continue to be fully supported as well.

For more information about this new device support, refer to section 8.2 of the "INTERACTIVE UNIX Operating System Maintenance Procedures" and the Section 7 manual entries for these drivers in the *INTERACTIVE UNIX System User's/System Administrator's Reference Manual*.

Support for the following devices has been significantly altered in Version 3.0:

- The High Performance Device Driver (HPDD) (previously referred to as the High Performance *Disk* Driver), which provides support for a wide range of disk controllers and SCSI host board adapters (HBAs) such as those listed above, has been enhanced to support machine configurations with large numbers of controllers/HBAs and attached devices. The old limits were two controllers/HBAs and four devices per controller. The new limits are six controllers/HBAs and 32 devices per controller. This provides support for many new high-end SCSI-based systems currently entering the market.

- Within the HPDD, support for SCSI tape devices has been enhanced significantly. For example, writing multi-volume tapes (for example, backups that span more than one physical tape) is now supported. See section 3, "SOFTWARE CORRECTIONS AND FIXES," of this document for a more detailed description of the problems that have been fixed.
- The floppy tape driver distributed in Version 2.2 has been removed from Version 3.0 and replaced by a driver that supports a wide range of Irwin floppy tape drive models.
- Tape support in general has been unified with the addition of a universal command, `mt`, for control (for example, erasing, retensioning, and so on) of any supported tape drive. This utility significantly eases the job of automatically managing tape backup via shell scripts when used with the "no-rewind" device. Refer to *mt*(1) for usage instructions.

The following device drivers are no longer included in Version 3.0, due either to obsolescence or to more current versions being available from third parties. Users upgrading from previous releases may still be able to run their existing versions of these device drivers under Version 3.0, but this is not recommended.

- The SunRiver* Fiber Optic Station drivers. The latest SunRiver device drivers may be downloaded free of charge from the SunRiver bulletin board. Call (512) 835-8082, with a 1200 or 2400 baud modem, 8 data, 1 stop, no parity bit ANSI is supported for a color display. If you have questions, contact SunRiver Technical Support at (512) 835-8001 (voice) or (512) 835-8026 (FAX), or send email to `uunet!sunriv!support` or `support@sunriver.com`.
- The UnTerminal* Station drivers. The latest versions of these drivers may be obtained directly from Advanced Micro Research, Inc. at (408) 456-9400. Drivers are available for the following types of UnTerminal Multi-Station adapters:
  - Video Network Adapter (VNA), a four-user Hercules* graphics adapter.

  - Video Network Adapter Plus (VNA Plus), an eight-user Hercules graphics adapter.
  - Video Graphics Network Adapter (VGNA), a two-user VGA graphics adapter.
  - Video Graphics Network Adapter Plus (VGNA Plus), a four-user VGA graphics adapter.
- The Bell Technologies ICC* Multiport Card driver.
- The Hub Multiport Serial Card driver.
- The DigiCHANNEL MC/Xi* Async driver.
- The INTERACTIVE MultiView DeskTop HSM host support module. This is not really a driver, but rather a module needed to support older versions of the MultiView DeskTop product; this module is now obsolete.

2.7 Very Fast File System

A new file system type, the Very Fast File System (VFFS), has been added to Version 3.0. As its name indicates, for certain types of file system usage it is significantly faster than INTERACTIVE's standard Fast File System, which stands on its own merits as one of the industry's fastest S5-compatible file systems. The programming interfaces of the S5 and VF file systems are identical, and thus when appropriate, the VFFS can transparently replace the S5 FS. The on-disk structure of the VFFS is not, however, S5-compatible, which means it cannot be used when interoperability with S5 file systems on other UNIX System platforms is required.

The VFFS is a mountable UNIX System file system that is designed for high-speed sequential reading and writing of very large files, such as files containing images. The transfer rate from disk approaches the maximum transfer rate of the drive itself. For example, on a 10 Mbit/sec ESDI drive, the transfer rate is roughly 700 Kbyte/sec, about twice as fast as the S5 file system. This performance is achieved by two techniques:

- Files are allocated in large, contiguous "extents."
- Whenever possible, data is transferred directly to and from the user program address space, rather than through the buffer cache.

Note that the VFFS is *not* intended as a replacement for the general purpose S5 file system. It should be used only in systems that are largely dedicated to applications (such as image or sound manipulation) with throughput requirements that are based on frequent movement of large data objects.

For more information on this new file system type, refer to section 5.6.1 in the "INTERACTIVE UNIX Operating System Maintenance Procedures."

2.8 CD-ROM File System

A new file system type, the High Sierra File System, has been added to Version 3.0. It supports two common file system formats for CD-ROM disks, the High Sierra and the ISO 9660.

By mounting a CD-ROM disk written in these formats using the standard `mount` command, files on the CD-ROM disk can be read as if they were part of the user's regular INTERACTIVE UNIX System file system. Of course, the CD-ROM file system cannot be written to.

This support allows applications to access a large base of information that is available through various sources on CD-ROM disks in these formats. For more information on this new file system type, refer to section 5.6.2 in the "INTERACTIVE UNIX Operating System Maintenance Procedures."

2.9 Miscellaneous Enhancements

- If you have more than one operating system loaded on your primary fixed disk, you can now select which one will be started any time you boot your machine. For example, if you have both an MS-DOS* (DOS) partition and an INTERACTIVE UNIX System partition, whenever you boot your machine a menu is displayed and you are prompted to select which partition should be booted. If you do not respond within 20 seconds, the "active" partition (as determined by your FDISK table) will be automatically booted, as in earlier releases. This new feature makes it easier to switch between different operating systems in different partitions on your fixed disk.

- The Fast File System (FFS) has been enhanced to provide a significant performance improvement when mounting and unmounting large file systems. This is particularly noticeable whenever a machine with a large fixed disk capacity is being booted or shut down; the process of mounting and unmounting all of the file systems can literally go from several minutes to several seconds.

- The version of the C shell (`csh`) shipping with Version 3.0 incorporates all fixes and upgrades from the Berkeley 4.3BSD "Tahoe" release, most notably fixes in the shell grammar and many efficiency improvements. For a list of the bugs that are fixed in this new version, refer to section 3 of this document, "SOFTWARE CORRECTIONS AND FIXES."
- The version of the mail router (`sendmail`) shipping with the INTERACTIVE UNIX Operating System Version 3.0 is based on Berkeley's Version 5.65. The most notable of the improvements is that it no longer uses lock files to prevent simultaneous delivery of mail messages; rather, it uses the kernel's file locking facilities, thus reducing the amount of overhead involved in processing mail messages. This change also eliminates problems caused by system crashes while a message is being handled. For a list of the bugs that are fixed with this new version, refer to section 3 of this document, "SOFTWARE CORRECTIONS AND FIXES." The various `sendmail` guides have been updated to reflect the new version.
- The INTERACTIVE UNIX Operating System kernel had previously been distributed as a small set of large object files, as is traditional for System V Release 3.2-based systems. This makes the job of distributing software updates very difficult, as even a small change to the kernel requires a major update, and it is difficult to prevent these updates from conflicting or overlapping. With Version 3.0 the kernel is now being distributed as a large set of small object files contained within a library. This change is completely transparent to the user, but allows INTERACTIVE Systems Corporation to support its users in a more efficient and timely manner.
- The spreadsheet used during system installation to allocate space for file systems has been altered so that it is far more intuitive and easy to use.

3. SOFTWARE CORRECTIONS AND FIXES

Over 100 bugs have been fixed in software and documentation for this release of the INTERACTIVE UNIX Operating System. The following sections represent a condensed list of the most important of the software fixes.

3.1 Hardware Driver Problems

3.1.1 SCSI Tape Driver Corrections and Enhancements

- The driver failed to correctly report the error status of requests to the SCSI tape drive. This driver now correctly reports end of tape, enabling large backups across multiple tapes. The driver also detects bad media and correctly reports when there is no tape in the drive. The system no longer panics or hangs when the tape is broken or breaks while executing a driver request.
- The driver has been enhanced to fully support requests for a "no-wait" tape device, that is, one that returns to the caller without blocking for the request to terminate.
- The driver improperly handled requests to the "no-rewind" device, thus enabling multi-volume backup on one tape.
- The driver now properly fails requests to erase a tape opened in read-only mode and prints an informative error message.
- The driver prints a warning message when a DC600A tape is written in 120 MB format on a 150 MB tape drive. Users would sometimes attempt to read these tapes in a 60 MB drive, not realizing that this device cannot read formats greater than 60 MB.
- The driver now properly handles piping of `write` requests between the `cpio` and `dd` utilities.
- `ioctl` requests to the tape driver (erase, retension, rewind, seek file mark, and write file mark) are now properly handled. The prior version would hang on requests to seek file mark(s).
- The tape driver now properly handles a request to read from a new blank tape, returning read error rather than causing the Archive 2150S to perform an endless series of failing reads.
- The tape driver now correctly refuses the request rather than hanging when it receives a request for I/O that is not a multiple of the tape drive's native block size.

3.1.2 `lp` (Printer Subsystem) Corrections

- If jobs were spooled to a printer that was off line, when the printer was turned back on, the jobs were not unspooled properly, and that printer's job queue was hung up until the next system reboot.
- For serial printers, setting `stty` options on the `lp` or `lpadmin` command line did not work.
- The `-o nobanner` option of `lp` did not work.

3.1.3 Miscellaneous Driver Corrections

- The limit on the size of a fixed disk attached to the system was 1 Gb. This has been extended and disks of arbitrarily large size are now supported.
- The system did not support DTC 7287 disk controllers (RLL format).
- When using a Future Domain SCSI host board adapter with more than one disk attached to it, heavy I/O to more than one of the disks at a time could cause the system to panic.
- On systems configured with an Adaptec 1542 SCSI host board adapter and multiple fast fixed disks, heavy I/O to more than one disk at a time would occasionally cause the system to hang.
- On systems configured with an Adaptec 1542 SCSI host board adapter, if a process attempted to read past the end of the UNIX System partition, the system would hang.
- Serial port transmission has been made more reliable when doing high-speed transmission (9600 baud and greater) on busy systems. In previous versions occasional loss of characters (up to five per thousand) could occur.
- Systems configured with a primary ST506 disk controller and a secondary ESDI disk controller could not access disks on the secondary controller.
- Certain valid VGA adapters, such as the ESD CMG, were not recognized as such due to the driver not masking off the 4 high bits of the color select register.
- I/O to the character device node corresponding to a disk on an Adaptec 1540 SCSI host board adapter would succeed even if the device was powered off.

- The Scroll Lock key, which works to stop and start screen output, did not light up as it should when output is stopped.
- If the system panicked with the kernel debugger configured into the system and the current virtual terminal was not the console, it was often impossible to get to the console where the debugger information was being displayed, particularly if a graphics application such as VP/ix* or INTERACTIVE X11 was in use on the current virtual terminal. The system now automatically switches to the console and returns to normal character mode. It also inhibits inadvertently entering the debugger from another virtual terminal using the CTRL ALT d keystroke combination when in graphics mode.
- The built-in Mouse Driver (supplied on the *Additional Drivers* diskette) did not support 3-button mice.

3.2 S5 File System Corrections

- Programs that created and removed large numbers of files, most notably the `news` application, could cause file system damage. Most of the problems that caused this symptom were fixed in Version 2.2, but one loophole remained that would allow it to be exhibited under very rare circumstances; this final loophole has been closed.
- Under very unusual circumstances, it was possible for the `root` file system to incur damage if the system was brought up after being shut down improperly.
- If the system discovered a bad block on a fixed disk, the system could panic in the process of printing out the console message informing the administrator of the bad block. Again, this happened only under rare conditions.
- If two directories under the same parent directory were links to each other, then removing one of those directories would ultimately result in file system damage.

3.3 `sysadm` System Administration Corrections

The new `sysadm` utility, in addition to providing a superior look-and-feel and much new functionality, also fixes a number of specific problems reported in the old `sysadm` utility. For example:

- `sysadm addbadblocks` would fail on the first disk of the second controller in a system.

- You could not update the number of ttys on a system using `sysadm` after installing the Multi-User Upgrade extension.
- Using `sysadm devicemgmt` (to manage UUCP devices), you could not enter uppercase letters as part of the device name.
- `sysadm umountfsys` did not work with DOS file systems or UNIX System file systems on low density diskettes.
- `sysadm installpkg` sometimes scrolled information off of the screen faster than it could be read.
- `sysadm lineset` could fail with varying symptoms depending on the system configuration.
- `sysadm` would reset the terminal's erase character to **CTRL** **h** regardless of what it was when `sysadm` was invoked. `sysadm addharddisk` would also reset the interrupt character to **DEL**.
- `sysadm addharddisk` would create file system directories with a default mode of 777, which allowed all users full access. It also did not allow proper flexibility in naming the new file systems, nor did it add them by default to `/etc/fstab` for automatic mounting on system boot.
- `sysadm adduser` would not allow you to specify an existing directory as the user's home directory, making it cumbersome to have users share home directories, or to have a user log in to a predefined and existing directory.

3.4 C Shell (`csh`) and Job Control Corrections

- The `%=` and `%-` commands did not work.
- The default job control switch character was improperly set such that it was possible to have it transmitted unexpectedly. The current system has the default switch character undefined, so job control is always disabled until an `stty` command is issued to set the switch character.
- `csh` did not get signal names right (for example, `kill -WINCH` *pid*).
- `csh` did not notice properly when a process dumped core.
- `csh` did not work properly with an NFS*-mounted directory in `$PATH` or in expanding names of files in NFS-mounted directories.

- Shell variables may now have longer names, and they may now contain numeric characters and underscores (_) in addition to alphabetic characters.
- Processes that attempted to catch `SIGCONT` would catch it properly the first time the signal occurred, but not subsequently.

3.5 `sendmail` (Mail Router) Corrections

- `sendmail` is generally much more stable when dealing with large volumes of messages, such as on a mail hub machine.
- The `alias` database was not properly locked while being updated, allowing multiple updates to corrupt the `dbm` files.
- Several bugs related to use of the name server (DNS/BIND) have been fixed, and better support for MX records is present.

3.6 `dossette` (DOS File System Access) Corrections

- Changing diskettes during a session could cause `dossette` to become confused and begin endless retries.
- Use of the `B:` drive (second diskette drive) was erratic under some circumstances.
- It was not possible to format a 3.25-inch diskette in 720Kb (low density) format.
- Diskettes formatted under `dossette` did not always interoperate properly with standard DOS or the VP/ix Environment.

3.7 Miscellaneous Corrections

- Processes running in POSIX mode (that is, compiled as POSIX applications) would sometimes cause a system panic.
- Systems equipped with Intel 80486 processors and configured with a very large number (5000, for example) of kernel buffers (tunable parameter `NBUF`) would occasionally panic.
- A process that attached the same IPC shared memory address more than once could cause the system to hang under heavy load.
- If a program placed a kernel record lock on a file in a DOS-FSS-mounted DOS file system and then did not release it before exiting, record locking on the system in general would be broken until the next system reboot.

- When using DOS-FSS file systems, if the parameter NDOSINODE, as defined in an obscure system configuration file, had a different value than the standard kernel tunable parameter NINODE, DOS-FSS would display erroneous behavior after some amount of use. The NDOSINODE parameter no longer exists and this problem no longer occurs.
- The `fdisk` command would occasionally produce a garbled display when dealing with disks with more than 1000 cylinders.
- During system installation it was not possible to define more than 7 file systems, even though the actual limit is 13.
- If the `/etc/TIMEZONE` file had a comment line before the setting of the `TZ` variable, the system would fail to boot. Also, if it had a legal time zone specifier greater than 11 characters long, the `su -` command would not work.
- If you chose a non-U.S. keyboard during system installation, after the system was installed and you rebooted, the keyboard behavior reverted to U.S. behavior.
- Setting the time zone during system installation did not work.
- Setting up `inittab` to come up in single-user mode by setting the `initdefault` line to an `s` did not work.
- The `tar` command could not be used on files whose path names were longer than 49 characters.
- The `pg` command would loop forever when used to page through very large files (well over 30,000 lines).
- The pseudo-tty driver (distributed on the *Additional Drivers* diskette) could not be configured into the kernel unless the INTERACTIVE TCP/IP or the INTERACTIVE X11 extension was also installed and configured.
- Using the `#!` feature to specify an alternate interpreter for the shell script would cause the process to hang if the path name to the alternate interpreter was more than 18 characters.
- The `TCSETAF ioctl` of the `tty` driver would cause the write queue to be flushed; only the read queue is supposed to be flushed on this `ioctl`.
- The permissions as distributed on some system directories and files were not as tight as possible to maximize security.

- Under some circumstances arrow keys would not work with the `vi` editor.
- If the `cu` command encountered line noise, it would lock and become an unkillable process.
- Going from multi-user to single-user mode and then back again resulted in multiple `cron` daemons running, resulting in `cron` jobs being run numerous times.
- If the password for a user whose login name was longer than eight characters expired, the user could not log in again, since it was impossible to change the password.
- It was impossible to remove the User's Manual Entries subset; although the removal appeared to work, the files remained on the fixed disk.
- The `terminfo` entry for `at386` incorrectly specified the `xt` option for destructive tabs as being supported.
- The `comm` command did not behave like the `sort` command in collating text with non-ASCII characters; they now both treat all characters as 8-bit quantities.
- The command `dfspace` would incorrectly report free space remaining on file systems that were completely out of space.
- The `inittab` file was limited to 100 entries; this has been increased to 400.
- The command `who am i` failed for users whose login name was eight characters long.
- The `nawk` command did not recognize command line variable assignments to variables with an underscore (_) in the name.
- The `stack -u` subcommand of the `crash` command did not work.

4. HARDWARE NOTES

The hardware base supported by the INTERACTIVE UNIX System continues to grow with Version 3.0 (refer to section 2.6, "Peripheral Support"). A large number of Intel-based PC platforms are supported in Version 3.0, including various manufacturers' machine models in combination with various disk controllers, tape drives, and so on. The definitive list of officially supported hardware is the INTERACTIVE UNIX System V.3.2 Hardware Compatibility Guide, available through the INTERACTIVE TeleServices Department at (800) 346-7111 or outside the United States, from your local distributor. In addition, with the large and ever-growing ranks of PC hardware, many platforms not explicitly named in the Compatibility Guide will also run the INTERACTIVE UNIX Operating System.

Due to this large hardware base, there are occasions where issues arise with particular manufacturers' units. The items mentioned here have been discovered in testing Version 3.0. Although these notes refer to specific manufacturers' equipment, they are meant to imply neither an endorsement nor a condemnation of either the manufacturer or the equipment. They are supplied so that users with this equipment know in advance about problems they are likely to encounter and workarounds where available.

4.1 Installation on EISA Disk Controllers

Particular care must be taken when installing the INTERACTIVE UNIX Operating System on EISA bus machines with EISA disk controllers or EISA SCSI host board adapters. Most of the newly-supported EISA controllers have both a compatibility mode, which emulates a particular ISA controller, and an enhanced (or extended) mode, which is their true EISA mode.

To take advantage of the speed and features provided by the EISA bus, these controllers must run in their enhanced mode. However, for many of these controllers, the INTERACTIVE UNIX System kernel that is provided on the *Boot* diskette supports only the compatibility mode. Machines with these controllers must be initially installed with the controller in its compatibility mode (note that in this mode, the kernel will only "see" the first 16 MB of the machine's physical memory, even if it has more). Then the HPDD subsystem must be reconfigured and the kernel rebuilt using the `kconfig` utility, the system must be shut down, the controller changed to run in enhanced mode, and the new kernel booted. The controller will then be fully supported running in its enhanced mode (and the kernel will "see" all of the machine's physical memory).

Switching the controller between compatibility and enhanced mode is an operation that varies depending on the controller; instructions should be provided by the controller hardware manufacturer. Details on which controllers must be installed in compatibility mode, and what the various hardware settings (IRQ, DMA, I/O addresses) each controller must be in during system installation, may be found in tables in section 8 of the "INTERACTIVE UNIX Operating System Maintenance Procedures."

4.2 Miscellaneous Notes

The following anomalies have been observed:

- The B-stepping of the Intel 80486/25 chip (this refers to a version of the 25MHz model of the 486 chip that is no longer in production, but is present in some existing machines) can cause significant problems for the INTERACTIVE UNIX System kernel. These problems can range in severity from a total inability to install the INTERACTIVE UNIX Operating System on machines with this chip, to occasional system hangs and panics under heavy loads on such machines. No software workaround has been discovered for this problem. Replacing the chip with a current stepping (C or later) causes the problem to disappear completely.

- Installation on machines with a DPT model 2011 or 2012 SCSI host board adapter requires some special steps in configuring the board. For the 2011, the I/O base address must be switched from its default factory setting to address 0x230; refer to the DPT documentation for instructions. For both controllers, the `dptfmt` utility supplied by DPT must be run. In doing so, make sure that the emulation information for both drives is set to `disabled` and that the drive types for both drives are set to `0`. This allows proper operation of these controllers in their enhanced mode, and ensures that the kernel can see all configured devices.

  When using `kconfig` to reconfigure the DPT 2012 to run in enhanced mode, make sure to set the I/O address and DMA channel to `NA` (not applicable); the `kconfig` default has them set to the DPT 2011 settings, since the same driver is used for both controllers.

The following firmware revisions are required for DPT boards:

DPT Board	*Firmware Revision*
2011	002F
2012/A	003G
2012/B	002G

In all cases, the Smart ROM version should be 1E and the EISA configuration file should be version 3B. Versions later than these should also work properly. Earlier versions will work to some degree, but are not recommended.

- The Adaptec AHA-1740 EISA SCSI host board adapter running in its enhanced mode occasionally displays an error message such as `a174x_diskint: Data overrun or underrrun`. This is caused by faulty firmware on the board and only occurs on firmware with a checksum of either 84DF or 865F. This checksum is located on the EEPROM at location U2 in the upper left-hand corner of the board, where there is a 7-digit part number followed by `CS` and then the checksum. If the checksum is not one of the two mentioned above, this problem will not occur. If it is one of the two, obtain and load updated firmware from Adaptec on to the board.

- The Western Digital WD-7000ASC SCSI host board adapter will fail when used with a Connor CP3100 SCSI disk drive if the firmware revision on the Western Digital board's U46 EPROM is earlier than version 62-000243-809. This is due to a problem with the firmware prior to this version.

- The driver for the various supported models of Future Domain SCSI host board adapters do not support SCSI tape devices with a native block size greater than 512 bytes, such as the Wangtek 5525 quarter-inch tape drive or the Archive* 4520 DAT.

- The Dell* 450SE machine cannot be installed with the INTERACTIVE UNIX Operating System unless the external cache is disabled. It is sometimes the case on various machine architectures that disabling of caches and shadow RAM areas can suddenly make "uninstallable" machines installable without any adverse effect once the machine has been successfully installed. This can be worth a try in the unlikely event that you find your 386/486 machine unable to boot the INTERACTIVE UNIX Operating System from the *Boot* diskette.

5. CAVEATS, PROBLEMS, AND WORKAROUNDS

The following deficiencies are known to exist in Version 3.0. Where available, workarounds are presented:

- If you are installing the INTERACTIVE UNIX Operating System for the first time on to a system currently running a non-INTERACTIVE version of a UNIX or XENIX* operating system, the installation procedure may fail. This usually only occurs when the INTERACTIVE UNIX System is being installed on to the old `fdisk` partition. In this case, reboot your system using your non-INTERACTIVE boot diskette and use the `fdisk` program to delete the existing UNIX System partition. Proceed according to the installation instructions.

- The new `sysadm` utility, when run on the system console screen, does not suppress messages printed to the console by the kernel or other system services, even when those messages are caused by the `sysadm` action you are performing. For example, if you are logged in to the console and attempt to format a diskette with `sysadm` and do not properly insert the diskette into the drive, the standard console message `FD(0): diskette not present -- please insert` will appear on your screen. Note that the screen can always be refreshed within `sysadm` by simply typing $\boxed{\text{CTRL}}$ $\boxed{\text{r}}$.

- The DOS-FSS facility cannot be used to mount DOS partitions that have been formatted using DOS versions 4.0 and later.

- Within the new `sysadm` and `kconfig`, occasional anomalies in screen appearance have been seen. For example, the main screen border can disappear and the color of menus can change from green to brown. These problems occur only rarely and are completely cosmetic in nature. On *extremely* rare occasions, fatal errors have been seen in the toolkit that these programs are based upon, causing a fatal exit from the program; should this occur, simply restart `sysadm` or `kconfig`.

- The first time `sysadm` is invoked, and whenever it is invoked after adding a package containing new `sysadm` scripts, the startup time is noticeably longer due to the need to rebuild various supporting databases.

- The names of subset and extension packages that appear on screen during installation via `sysadm installpkg` are truncated to 58 characters. A few package names exceed this limit.

- Some problems exist in the `sysadm backup` procedures:
  - Only one user at a time on the system may run a backup procedure. No check is made for this condition.
  - The `sysadm backup` menu, `Back Up Selected Files`, has a verify option. It only verifies the last volume of a multi-volume backup. As a workaround, earlier volumes can be verified manually using the `dd` or `cpio` commands.
  - On backups to tape, make sure to specify the "rewind" tape device name, as specifying the "no-rewind" device causes the backup to fail.
  - Backups made with an absolute path name can only be restored to their absolute locations, even if a directory to restore to is specified using the `restore` option. It is recommended that backups always be made using relative path names (that is, without the leading `/`).
- If the background and foreground colors on the screen were changed via the `setcolor` command, they will be changed back to the default when using the `sysadm` command.
- When the `kconfig` command is used to reconfigure the HPDD, it does not default to the old settings present the last time the HPDD was reconfigured. Rather, you must fully specify the characteristics of your fixed disk controllers (how many, what attached devices, what IRQ level, and so on) each time you choose this option.
- System accounting information on fixed disk performance is not maintained, making the `sar -d` command nonfunctional. This does not impair the utility of the system accounting package as a whole.
- Installation of some subsets, extensions, and third-party software can occasionally produce the error message `? /etc/conf/kconfig.d/description`. This may be safely ignored.
- Users of the International Supplement should note that the XPG3 version of the `tar` command produces archives that cannot be read by older versions of `tar` (as per the XPG3 standard). If such backward compatibility is required, the standard System V `tar` command, which is saved after loading the International Supplement, should be used.

6. INSTALLING VERSION 3.0

If you are updating your system from a release of 386/ix* (2.0.2 or earlier), it is necessary to perform a complete (destructive) installation. Make sure you have backed up all of your user files, and those system files that you have altered (such as `/etc/passwd`), before performing such an action. You may then restore these files on to your newly installed INTERACTIVE UNIX System Version 3.0. Whether you are installing the INTERACTIVE UNIX System on a machine for the first time or performing a destructive update from the 386/ix Operating System as described above, follow the instructions in the "INTERACTIVE UNIX Operating System Installation Instructions" to do a complete installation.

If you are updating your system from an earlier installation of the INTERACTIVE UNIX Operating System (Versions 2.2 or 2.2.1), it is not necessary to perform a complete reinstallation of your system. However, if you want to *change* the disk partitioning of your *primary* disk, you cannot use the update option; you *must* reinstall the entire system (destructive installation).

If you choose to perform a nondestructive installation, you must also reinstall all optional subsets and extension packages that are installed on your current system. Most of these packages have been updated since the earlier 2.*x* releases. The following files are saved (if they exist) and restored to an *updated* system:

```
/.profile
/etc/conf/cf.d/init.base
/etc/fstab
/etc/group
/etc/inittab
/etc/partitions
/etc/passwd
/etc/shadow
/etc/TIMEZONE
/etc/bupsched
/etc/gettydefs
/etc/ttytype
```

```
/etc/issue
/etc/profile
/etc/default/login
/etc/default/passwd
/etc/default/su
/usr/spool/cron/crontabs/adm
/usr/spool/cron/crontabs/root
/usr/spool/cron/crontabs/sysadm
```

System files not listed here will be overwritten during the update installation. If you have modified other system files, back them up before proceeding. Be wary of restoring the older versions of system files, since in some cases only the newer versions will work correctly.

Before updating your INTERACTIVE UNIX Operating System, you must ensure that your primary fixed disk is correctly identified in the file `/etc/partitions`. The installation procedure requires this disk to be named `disk0`. This is the default name, but if you have edited the `/etc/partitions` file or have reconstructed this file using the `mkpart -x` command, it may have a different name. The `/etc/partitions` stanza for the primary disk contains the string `device = "/dev/rdsk/0p0"`. Check to see that this stanza is labelled `disk0:` and change it if necessary.

To update the INTERACTIVE UNIX Operating System, shut down your system and insert the Version 3.0 *Boot* diskette. Read the first three sections of the "INTERACTIVE UNIX Operating System Installation Instructions." Then follow the instructions in section 4. When the main installation menu appears, select the `Update` option and follow the instructions on the screen. Note that making a full system backup before performing an update is strongly recommended.

After updating, if you have more than one disk controller and/or SCSI host board adapter in your system, the device names in your `/etc/fstab` file for file systems on the second controller will be wrong. If the second controller is non-SCSI, change names of the form `c1d`*n*`s`*n* to `c1t`*n*`s`*n*, where *n* is a digit. If the second controller is a SCSI, the digit associated with the first *n* should be changed to match the SCSI target ID of the disk that file system is on.

6.1 Updating Optional INTERACTIVE Subsets and Extensions

If you have a version of the Kernel Configuration subset (kc) already installed on your machine, you must reinstall the 3.0 version. The following files will be saved and restored during the update:

- /etc/conf/cf.d/mdevice
- /etc/conf/cf.d/stune
- /etc/conf/kconfig.d/description
- All files in /etc/conf/sdevice.d
- All files in /etc/conf/sfsys.d

If you have installed additional device drivers, they should not be affected by updating the kc subset. However, inittab entries or startup scripts could be affected.

You must also reinstall all of the drivers from the *Additional Drivers* diskette that you are currently using. Refer to Appendix A for a list of the drivers on this diskette.

After reinstalling the kc package, use kconfig to build a new kernel with the Version 3.0 fixes and drivers installed. Note that you must also choose the Reconfigure HPDD option of kconfig at this time, as your old HPDD configuration cannot be saved.

As with the Kernel Configuration and Additional Drivers subsets, you should update all subsets that you have installed on your existing machine with the Version 3.0 counterparts to ensure that you bring these subsets up to a Version 3.0 level of functionality. There is no guarantee that an earlier version of any subset will function properly under Version 3.0 of the operating system, as such combinations have not been tested. Note, however, that this statement does *not* apply to applications software from INTERACTIVE or third parties, for which binary compatibility with earlier versions of the INTERACTIVE UNIX Operating System has been carefully preserved. Such packages should run unchanged after updating your system to Version 3.0.

Appendix A: SUMMARY OF INTERACTIVE SUBSETS

The INTERACTIVE UNIX Operating System is supplied with a number of software subsets. Depending upon your individual requirements and the size of your fixed disk, you should select and install only those subsets that are necessary for your daily use.

The *Boot*, *Install*, and *Core* subset diskettes are required to install and run the INTERACTIVE UNIX Operating System. The remaining subsets are optional. All subsets that do not appear on any diskette label are located on the *Additional Drivers* diskette.

The following tables show the purpose of each subset and the *approximate* amount of fixed disk storage that is required.

Required Subsets

Subset Name and Function	*Storage Required*
Core The Core subset contains the most commonly used system commands and utilities. It also contains the utilities that are required to drive system peripherals (such as printers) and to maintain the system.	7 MB

Optional Subsets

Subset Name and Function	*Storage Required*
Basic Networking The Basic Networking subset contains the files and utilities required to configure and use the standard UNIX System networking utility (`uucp`) and network mail router (`sendmail`).	1.4 MB
File Management The File Management subset contains the commands and utilities used to manipulate UNIX System files.	0.4 MB
Kernel Configuration The Kernel Configuration subset contains the programs and configuration files required to configure and build a new UNIX System kernel.	4.0 MB
2 Kilobyte File System Utility Package The 2 Kilobyte File System Utility Package subset provides an optional method of file system organization employing larger (2K) block sizes to alter disk input/output (I/O) performance.	0.2 MB
XENIX File System Package The XENIX File System Package subset provides support for mounting and using XENIX file systems just as you would under XENIX System V.	0.1 MB
High Sierra File System The High Sierra File System subset provides support for the most popular formats of file systems on CD-ROM disks.	0.1 MB
Very Fast File System The Very Fast File System subset provides an alternative file system that can provide significant performance gains when used for storing very large files, such as those storing images.	0.3 MB

Optional Subsets	
Subset Name and Function	*Storage Required*
Help Utilities The Help Utilities subset contains the commands and utilities used to provide a simple on-line help facility.	0.6 MB
Spell Utilities The Spell Utilities subset contains the file spelling checker facilities.	0.2 MB
Terminal Utilities The Terminal Utilities subset consists of the `terminfo` database that contains the descriptions and operating capabilities of over 150 popular terminal devices and terminal filters that allow a variety of terminals to print formatted output.	0.7 MB
User's Manual Entries The User's Manual Entries subset contains all the user and system administration commands and special files.	1.0 MB
STREAMS Facilities The STREAMS Facilities subset contains networking commands and the drivers needed to run packages that use STREAMS.	0.4 MB

Optional Subsets	
Subset Name and Function	*Storage Required*
Pseudo-TTY Drivers The Pseudo-TTY Drivers subset is required by many system extension packages (such as INTERACTIVE X11 and INTERACTIVE TCP/IP) to enable the establishment of login sessions that are not associated with a physical device, such as a console or a serial port.	0.1 MB
Archive Cartridge Tape Driver The Archive Cartridge Tape Driver subset provides support for various models of Archive cartridge tape devices.	0.1 MB
Wangtek Tape Controller Driver The Wangtek Tape Controller Driver subset provides support for various models of Wangtek cartridge tape devices.	0.1 MB
Logitech Bus Mouse Driver The Logitech* Bus Mouse Driver subset provides support for the Logitech Bus Mouse pointing device.	0.1 MB
Microsoft Mouse Driver The Microsoft* Mouse Driver subset provides support for the Microsoft Bus Mouse pointing device.	0.1 MB
Built-In Mouse Driver The Built-In Mouse Driver subset provides support for various on-board pointing devices such as those on many COMPAQ models and IBM PCs.	0.1 MB
COMPAQ 525MB SCSI Tape Driver The COMPAQ 525MB SCSI Tape Driver subset provides support for the COMPAQ 525MB SCSI tape device.	0.1 MB

Optional Subsets	
Subset Name and Function	*Storage Required*
Irwin Floppy Tape - Mini Cartridge The Irwin Floppy Tape - Mini Cartridge subset provides support for various models of Irwin floppy tape devices.	0.4 MB

Appendix B: VERSION 2.2.1 INFORMATION

1. INTRODUCTION

INTERACTIVE UNIX System V/386 Release 3.2, Version 2.2.1 incorporates a number of significant bug fixes and one important new feature into the INTERACTIVE UNIX System Version 2.2 product. These release notes supplement the Version 2.2 release notes and describe only the differences between Version 2.2 and Version 2.2.1.

2. DATA COMPRESSION

The new feature incorporated into Version 2.2.1 is data compression on the distribution media. The files on the distribution diskettes are in compressed format and are decompressed during installation. The entire process is transparent to the user; the interface is essentially identical to that in Version 2.2.

What is *not* transparent, however, is that most of the operating system subsets are now distributed on fewer diskettes, which significantly decreases the time it takes to install the INTERACTIVE UNIX Operating System. Most of the other products in the INTERACTIVE Product Family will soon be distributed in compressed format, resulting in similar savings in installation time. For both the operating system product and for the product family as a whole, it is estimated that the number of diskettes and resulting installation times will be decreased on average about 30 to 40 percent.

3. SOFTWARE CORRECTIONS AND FIXES

The following significant bugs present in Version 2.2 have been fixed in Version 2.2.1:

- Inability to install on fixed disk drives with over 100 bad sectors or over 43 independent clusters of bad sectors. In practice, this meant that very large fixed disk drives were frequently not usable with Version 2.2.
- Incompatibility with binaries which use the *msgctl*(2) system call, when those binaries were produced on other SVID-compliant operating systems.

  ☛ Any application developer who has compiled an application under Version 2.2 should ensure that it does not use *msgctl*(2); if it does, that application should be remade under Version 2.2.1 using the updated `msg.h` header file found in the Kernel Configuration subset.

- During installation, attempting to change the default fixed disk parameters as determined by the installation software had no effect. One result of this problem was loss of usable disk space if one had to use a BIOS disk type that was smaller than the actual disk geometry.
- Inability to install on Adaptec 1640 SCSI controllers on some Micro Channel* bus computers.
- Inability to install on computers with Seagate IDE fixed disk drives, such as the model 157A.
- Inability to install on computers with Rodime IDE fixed disk drives, such as models RD3128A and RD3058A.
- Inability to install on DEC* Workstation 325 computers.
- Inability to install on computers with Western Digital 1007V ESDI fixed disk controllers.
- A number of minor problems inhibiting strict POSIX compliance, as evidenced by a few failures in the NIST PCTS test suites. PCTS Version 1.0 runs flawlessly on Version 2.2.1.
- Attempting to rebuild a kernel during installation (before rebooting off of the fixed disk) often failed with a mysterious error message.
- Inability to install if, during the disk preparation installation screen, one chose to do a full read/write surface analysis but did not choose to add extra defect information.

INTERACTIVE UNIX Operating System Installation Instructions

CONTENTS

INTERACTIVE UNIX* Operating System Installation Instructions

1. INTRODUCTION

This document describes the basic requirements and tasks that are necessary to initialize, install, and log in to the INTERACTIVE UNIX Operating System. It is for both new and experienced users of the INTERACTIVE UNIX Operating System who are responsible for the initial installation of the system.

The installation procedure is designed to be as self-explanatory as possible, with easily accessible on-line help. This document provides more information than is available in the on-line help facility and gives additional information necessary for users with nonstandard hardware configurations or special system needs. (The screens shown in this document are for purposes of illustration. Your actual screens may vary slightly.)

If you are new to the INTERACTIVE UNIX Operating System and other UNIX-based systems, you should read and understand the information presented in the "INTERACTIVE UNIX Operating System Primer" in the *INTERACTIVE UNIX System Guide for New Users* before attempting to install the INTERACTIVE UNIX Operating System. Once you have completed the basic system installation, refer to "System Administration for New Users of the INTERACTIVE UNIX Operating System" in that same book to learn how to install user accounts, back up and maintain files on the system, install and configure printers and other hardware devices, and tailor the system to match your requirements. **Experienced users** can refer to the "INTERACTIVE UNIX Operating System Maintenance Procedures" for more technical information.

1.1 Overview of This Document

This document is divided into eight major sections:

1. INTRODUCTION
   This section provides a general overview of this document.

2. GETTING STARTED
   This section outlines the minimum hardware requirements and some configuration information you need to install the INTERACTIVE UNIX Operating System.

3. USING THE INTERACTIVE UNIX SYSTEM INSTALLATION PROCEDURE
   This section describes how to use the menus, forms, and on-line help that make up the INTERACTIVE UNIX Operating System installation program.

4. INSTALLING THE INTERACTIVE UNIX OPERATING SYSTEM
   This section explains how to boot the system and load the operating system software. It provides a step-by-step example of an uncomplicated installation for new INTERACTIVE UNIX System users. It also describes how to back up your *Boot* diskette and how to install non-INTERACTIVE UNIX System partitions on the fixed disk.

5. SHUTTING DOWN AND REBOOTING THE SYSTEM
   This section describes how to initiate an orderly shutdown of the system and how to reboot the system after the computer has been turned off.

6. INSTALLING OPTIONAL SOFTWARE
   This section describes how to install optional software subsets or separately purchased packages, including XENIX* software.

7. Appendix A: FIXED DISKS AND CONTROLLERS
   This appendix describes the physical components of a fixed disk, interface types, the compatibility of various disks and controllers with the INTERACTIVE UNIX System, and information you will need if you plan to use multiple disks and controllers on your system.

8. Appendix B: THE HARDWARE SETUP PROGRAM
   This appendix gives you the information you need to set up a machine so that the INTERACTIVE UNIX Operating System can be installed.

1.2 How to Use This Document

- **If you have a new machine that has been set up by the vendor or you have a machine that is already set up and running an operating system but has never had the INTERACTIVE UNIX Operating System installed on it before**, start with section 2, then read this document in order.
- **If you have a new machine and you are doing your own setup** (i.e., the vendor is not setting the machine up for you), read section 2, "GETTING STARTED," then read Appendix B, "THE HARDWARE SETUP PROGRAM." Then return to section 3, "USING THE INTERACTIVE UNIX SYSTEM INSTALLATION PROCEDURE," and continue on through this document.
- **If you have a machine that is already set up and running a release of the INTERACTIVE UNIX Operating System**, start with section 3, "USING THE INTERACTIVE UNIX SYSTEM INSTALLATION PROCEDURE," and read this document in order.

1.3 What Will I Learn From This Document?

This document provides step-by-step instructions to help you install the INTERACTIVE UNIX Operating System for the first time. If you have never used an INTERACTIVE UNIX System or another UNIX-based System before, these installation instructions should provide you with all the necessary information you need. If you are an experienced UNIX System user, you can refer to them primarily when you need more detail. This document describes:

- The minimum hardware requirements for running the system.
- The basic installation procedures (most are performed automatically by the system).
- The on-line menus and forms and the help facility for installation.
- The first-time setup procedures.
- How to turn off your system.
- How to restart your system.
- How to back up your *Boot* and *Install* diskettes.
- How to install optional software subsets and extensions.

This document does not attempt to cover all of the installation and configuration options that are available for the INTERACTIVE UNIX Operating System. It does not attempt to explain how to install and configure networking options or other optional packages.

2. GETTING STARTED

Before you install the INTERACTIVE UNIX Operating System on your computer, you (or someone, such as your vendor) must complete the following tasks:

1. Assemble and set up your hardware.
2. Read this section to verify your hardware configuration and to determine what you need to do to configure your hardware to support the installation of the INTERACTIVE UNIX System.
3. Run the *setup* program supplied by your computer manufacturer (if either you or your vendor has not already set up your machine).
4. Determine your partitioning requirements and back up any existing partitions you plan to move or delete. You must also back up the partition onto which you plan to install the INTERACTIVE UNIX Operating System.

This section explains the specific hardware information you need to install the INTERACTIVE UNIX System.

2.1 Hardware Requirements

To successfully install and use the INTERACTIVE UNIX Operating System, in addition to the basic 386* or 486* AT*, EISA, or Micro Channel* architecture platform, you must have the following hardware components:

- **RAM**
  4 megabytes (MB) of 32-bit Random Access Memory (RAM) are required.
- **A fixed disk**
  One fixed disk of at least 40 MB capacity is required. A 40–80 MB or larger fixed disk is strongly recommended, and certain ARCHITECH* Series products may require additional storage capacity.
- **A fixed disk controller**
  A fixed disk controller or SCSI host adapter supported by the INTERACTIVE UNIX System is required. For a full list of supported controllers, refer to the INTERACTIVE UNIX Hardware Compatibility Guide, which can be obtained by calling the INTERACTIVE Systems Corporation Teleservices Department at (800) 346-7111.

- **A diskette drive**
  A high-density diskette drive that supports 5 ¼ inch or 3 ½ inch media is required.

- **A display controller and monitor**
  A monochrome, Hercules*, CGA, VGA, or EGA display adapter and monitor, or other display that correctly emulates one or more of those video standards, is required.

Compare these requirements with your hardware configuration. Make sure that you have available at least the minimum configuration. Then, if your hardware vendor has not already assembled and set up your system, follow the manufacturer's instructions to determine how to assemble and attach all components for operation and use the *setup* program supplied by the hardware manufacturer to configure your fixed disk and fixed disk controller. Refer to Appendix B for the INTERACTIVE UNIX Operating System requirements that you must consider when you configure your system using the hardware manufacturer's *setup* program.

Both basic and more advanced information about fixed disks and controllers is located in Appendix A. The remaining hardware components (RAM, diskette drives, display controller, and monitor) will not be discussed further, since the installation procedures provided with the INTERACTIVE UNIX Operating System require no information about them.

2.2 Fixed Disk Partitions

The INTERACTIVE UNIX Operating System allows you to divide the fixed disk into multiple "logical disks" called *partitions*. Multiple partitions enable you to have more than one operating system resident on a single disk. For example, you may use one partition to store the UNIX System and another partition to store MS-DOS* (DOS) or OS/2*. Your fixed disk may already be divided into several partitions.

Before you install the INTERACTIVE UNIX System, you must determine your partitioning requirements. If you plan to use multiple operating systems (DOS, XENIX, and the INTERACTIVE UNIX System, for example), you will require one partition for each operating system. The VP/ix* Environment does not require a separate partition for installation; it resides on the INTERACTIVE UNIX System partition. Before you install the system, you should consider how you plan to use each operating system you intend to install. You will want to allocate more disk space for the operating system that you will be using most frequently.

☛ If you want to change your partitioning later, you will need to perform a destructive (complete) installation.

Remember that when you determine your partitioning requirements, you must establish one partition of at least 20 MB in which to install the INTERACTIVE UNIX Operating System. You will establish this partition during the installation process (see section 4.3.3).

2.3 Backing Up Partitions

If you do not delete, move, or change any existing partitions on your fixed disk (and you do not reformat your disk), then installing the INTERACTIVE UNIX Operating System on your machine will destroy only the data in the partition in which you are installing the INTERACTIVE UNIX System. However, it is *always* a good idea to back up the partitions on your fixed disk, even if you do not plan to alter them.

☛ Deleting, moving, or changing any existing partitions on your fixed disk will cause *all the data in those partitions to be destroyed by the installation.* You should back up any data in such partitions *before* you proceed with the installation. Reformatting the disk during installation *will destroy ALL the data on the disk.* Back up everything you want from your disk if you plan to reformat it.

To back up data that currently reside on your fixed disk onto diskettes or tape, use the backup facilities available with your existing operating system (e.g., DOS, XENIX, UNIX, etc.). Use the `BACKUP` command on DOS and the `backup` or `dump` command on XENIX. Use the `sysadm` backup procedure on the INTERACTIVE UNIX Operating System.

Certain copy-protected DOS applications may require special backup procedures. If necessary, refer to the documentation supplied with your operating system or application for additional instructions on how to back up the data.

2.4 Backing Up Your Boot and Install Diskettes

You need the two diskettes labeled *Boot* and *Install* to bring up your INTERACTIVE UNIX Operating System. These contain a bootable UNIX System file system that could be corrupted if a system crash occurs. The *Boot* diskette is write-protected. However, the *Install* diskette cannot be write-protected and could become unusable in the event of a system crash. It is *strongly recommended* that you at least make a backup copy of the *Install* diskette.

2.4.1 Using 5¼ Inch Diskettes for Backup

If your system was delivered on 5¼ inch diskettes and you have a copy of MS-DOS version 2.0 or higher, you can use its `diskcopy` program to make a backup copy of the INTERACTIVE UNIX System *Boot* and *Install* diskettes.

1. If you have MS-DOS installed on your fixed disk, simply type the following command:

   ```
   DISKCOPY  A:  A:
   ```

2. If you have a copy of MS-DOS on a diskette, insert the DOS diskette into drive A and type the following command:

   ```
   DISKCOPY  A:  A:
   ```

   In either case the `DISKCOPY` program will prompt you through the copy process. Repeat for the *Install* diskette.

If one of these versions of MS-DOS is not available, you can use the INTERACTIVE UNIX System `sysadm` facilities to back up the *Boot* and *Install* diskettes after completing the INTERACTIVE UNIX System installation.

1. Once the system is installed, read section 3 of the "INTERACTIVE UNIX Operating System Maintenance Procedures" to learn about the system administration (`sysadm`) utilities. Then you can use the `cpdisk` option from the `Diskette Management` menu under `Disk` to back up your *Boot* and *Install* diskettes.

2. Or, you may use the `dd` command rather than the `sysadm` facility, as follows. Put in the *Boot* diskette and type the command:

   ```
   dd  if=/dev/rdsk/f0q15dt  of=/tmp/bootflop  bs=30b
   ```

   Remove the *Boot* diskette, insert a blank, formatted high density diskette, and type the following command:

   ```
   dd  if=/tmp/bootflop  of=/dev/rdsk/f0q15dt  bs=30b
   ```

   Repeat using the *Install* diskette.

2.4.2 Using 3½ Inch Diskettes for Backup

If you need to copy 3½ inch media, you cannot use the MS-DOS `DISKCOPY` utility. Use the `sysadm cpdisk` utility or the INTERACTIVE UNIX System `dd` command described above to back up the *Boot* and *Install* diskettes after completing the INTERACTIVE UNIX System installation.

☛ Note that you must use an `18` rather than a `15` in the command lines for 3½ inch media, that is, you will use `/dev/rdsk/f0q18dt`, *not* `/dev/rdsk/f0q15dt`.

See *dd*(1M) for more information about the `dd` command.

3. USING THE INTERACTIVE UNIX SYSTEM INSTALLATION PROCEDURE

The INTERACTIVE UNIX Operating System has an easy to use and understand installation procedure. It is designed so that all the information you need to perform an uncomplicated installation is available on-line. Although it may be helpful, you do not need to read this section before installing the INTERACTIVE UNIX Operating System on your computer. You can safely explore the system on-line; no destructive actions occur unless you are warned and then confirm that the action should take place. The installation can be stopped at any time by pressing CTRL \ (backslash) and then typing `shutdown`. You can restart the installation by typing `INSTALL` and pressing ENTER. If you need to refresh the screen at any time during the installation, press CTRL r and the screen will be refreshed the next time it changes.

This section discusses the menus and forms that make up the installation procedure. It also details how to get on-line help when you are not sure what you should do or if you just want more information about a particular part of the process.

The system is largely self-explanatory. For each part of the process, there is an introductory screen. Introductory screens either **1**) explain something about the information being requested and tell you why it is needed or **2**) explain what is happening to the system at this point during the installation. Experienced users who are familiar with the INTERACTIVE UNIX System installation procedure can turn off the introductory screens.

3.1 Using Help

Two levels of help are available at almost any time during installation: context-specific help and general information found in the Help Index.

3.1.1 Context-Specific Help

Context-specific help is associated with a particular question, menu option, or form field. You obtain context-specific help by pressing the help key, F1, when the cursor is on an item that you want to know more about. Whenever you are not sure what to do or what is being asked of you, press F1. If no context-specific help is available, the Help Index appears instead.

For example, one of the installation forms asks the following question:

```
Will the system be used
as a file server?            N
```

If you press F1 while the cursor is on this field, the system displays:

```
+-----------------------------File Server-------------------------+
|If you plan to use your computer primarily to contain            |
|shared file systems in a network environment, such as            |
|INTERACTIVE TCP/IP or INTERACTIVE NFS, you will need             |
|more swap space than a computer that is not used in a            |
|network.  Answer "y" to this question.                           |
+-----------------------------------------------------------------+
```

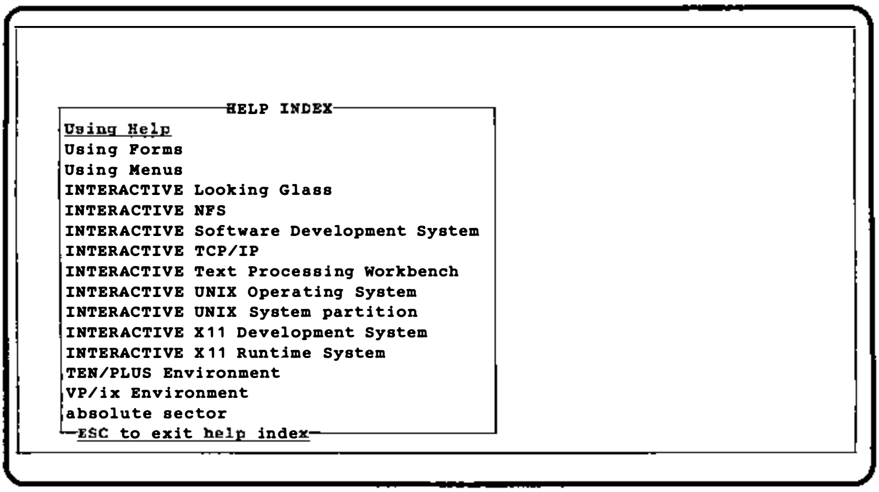
You can press any key to refresh the screen and continue with the installation.

3.1.2 The Help Index

Users who want more information about a topic than the help key provides can access a list of help topics, called the Help Index, by pressing F1 a second time. Words that are highlighted on the screen as you move through the installation procedure have corresponding entries in the Help Index.

The Help Index is a list of topics that pops onto your screen when you press F1 twice. (If context-specific help is not available, then the Help Index appears the *first* time F1 is pressed.) The list is alphabetical, except for the first three topics, `Using Help`, `Using Forms`, and `Using Menus`. These topics contain a condensed version of this section, which is important for users to read if they plan to install the INTERACTIVE UNIX Operating System without reading this entire document.

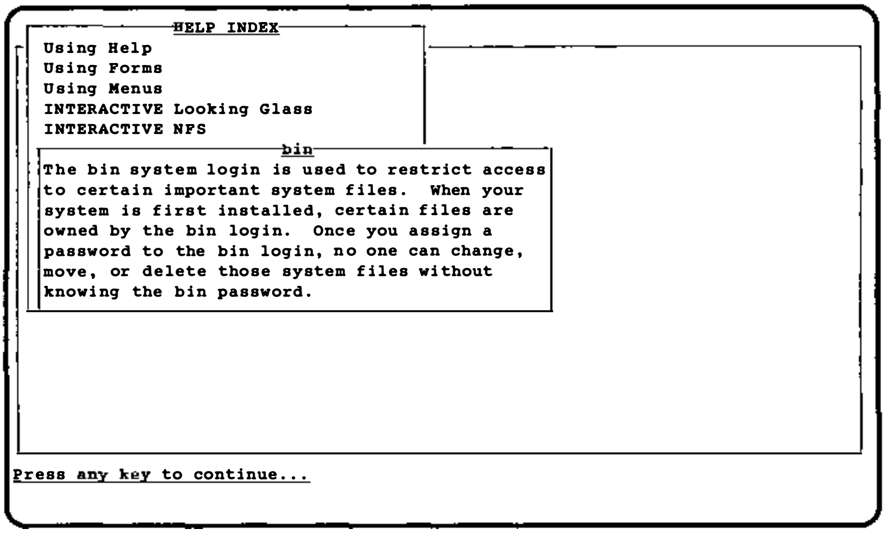
When you press F1 the second time, your screen looks similar to this:



```
HELP INDEX
Using Help
Using Forms
Using Menus
INTERACTIVE Looking Glass
INTERACTIVE NFS
INTERACTIVE Software Development System
INTERACTIVE TCP/IP
INTERACTIVE Text Processing Workbench
INTERACTIVE UNIX Operating System
INTERACTIVE UNIX System partition
INTERACTIVE X11 Development System
INTERACTIVE X11 Runtime System
TEN/PLUS Environment
VP/ix Environment
absolute sector
ESC to exit help index
```

The currently active item is highlighted. (Highlighting on the screens is represented here by underlining.) You may use the up and down arrow keys and PAGE-UP and PAGE-DOWN to look through the Help Index. Move to an option using the arrow keys, or press enough of the first characters in the option's name to distinguish it from the other items in the list. For example, if you type `c`, the cursor will move to `checkfsys`; if you want to move immediately to `conventions`, type `co`. The cursor will move immediately to `checkfsys` and then drop down to `conventions`.

Select an option by pressing ENTER. When you select an option, explanatory text is displayed in a box. For example, if you select `bin`, the system displays the following:



```
HELP INDEX
Using Help
Using Forms
Using Menus
INTERACTIVE Looking Glass
INTERACTIVE NFS

bin
The bin system login is used to restrict access
to certain important system files.  When your
system is first installed, certain files are
owned by the bin login.  Once you assign a
password to the bin login, no one can change,
move, or delete those system files without
knowing the bin password.

Press any key to continue...
```

Press any key to return to the Help Index. Exit the Help Index by pressing ESC.

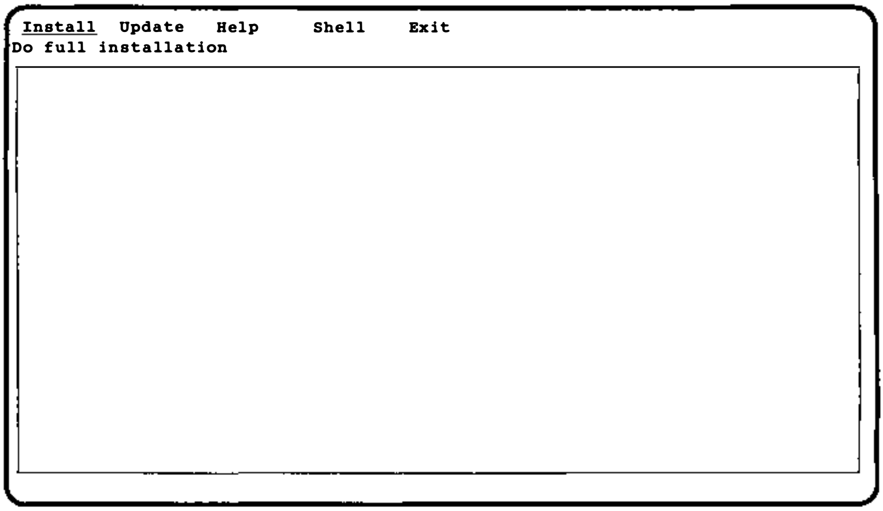
(Note that if you are installing on a portable computer or a computer with an 84-key keyboard, you will have to press the NUM-LOCK key after booting to enable the cursor motion, page-up/page-down, home, and end functions of the numeric keypad.)

3.2 Using Menus

A menu is displayed whenever you have to make a choice among different options. There are two type of menus: bar menus and pop-up menus.

3.2.1 Bar Menus

The first menu you see during installation is a bar menu.



```
 Install  Update   Help     Shell    Exit
Do full installation
```

Bar menus appear at the top of the screen. The currently active option in a menu is highlighted. Use the arrow keys to move between the menu options. As you move to each option, a brief description of that option appears under the menu. Additional, context-specific information is available for each menu option when you press F1. If you want more information after reading the context-specific help, press F1 a second time to display the Help Index and look through the index to see if there is a related entry.

Move to an option using the arrow keys or press enough of the first characters in the option's name to identify it. For example, on the installation menu:

```
Install    Update    Help     Shell     Exit
```

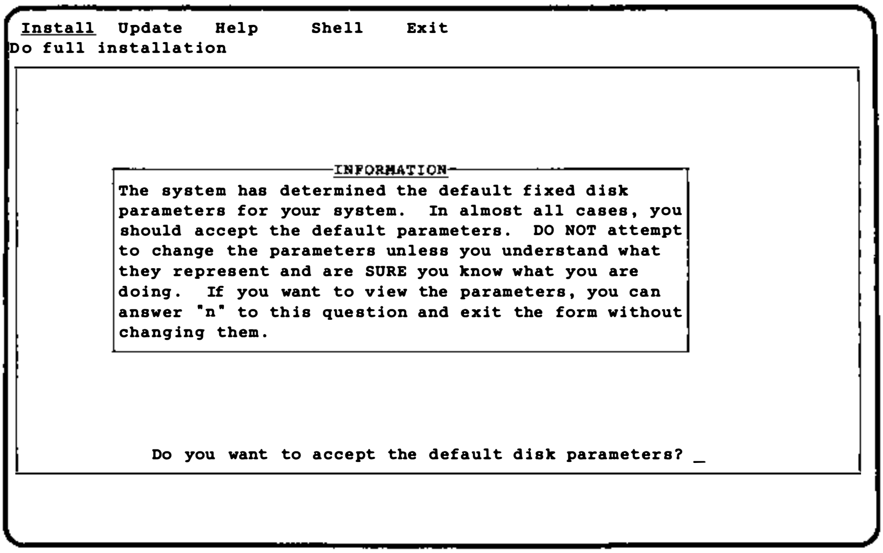
you can move immediately to the `Help` option by typing `h` or `H`. Select an option by moving to it and pressing ENTER.

3.2.2 Pop-Up Menus

In a pop-up menu, such as the Help Index, you can use the arrow keys or PAGE-UP and PAGE-DOWN to move from option to option. The currently active option is highlighted. Move to an option using the arrow keys or by pressing enough of the first characters in the option's name to uniquely identify it. Select an option by moving to it and pressing [ENTER]. To exit from a menu (or in some cases, to return to a previous menu), press [ESC]. Examples using a pop-up menu are presented in 3.1, "Using Help."

3.3 Using Forms

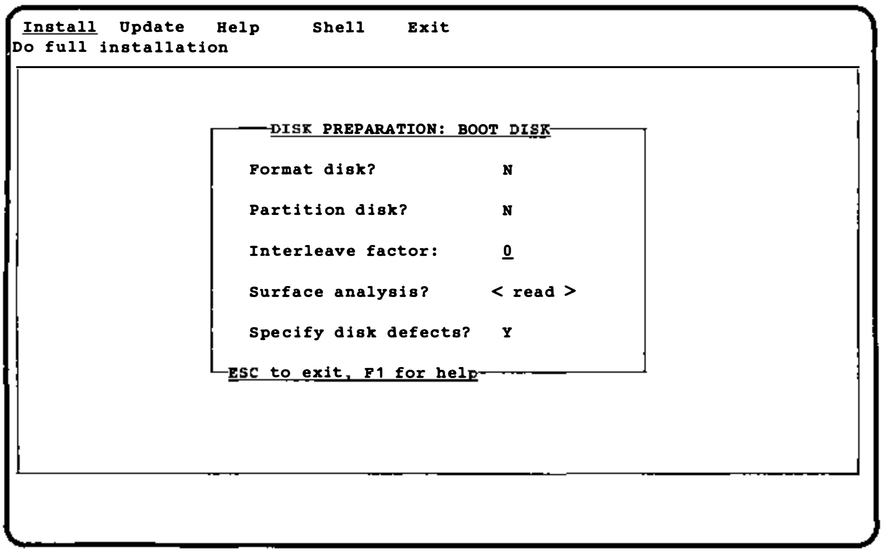
Whenever this installation procedure requires you to provide information, a form is displayed. Some forms consist of just one question, which is answered yes or no. For example:



```
 Install  Update  Help    Shell   Exit
Do full installation

                         INFORMATION
          The system has determined the default fixed disk
          parameters for your system.  In almost all cases, you
          should accept the default parameters.  DO NOT attempt
          to change the parameters unless you understand what
          they represent and are SURE you know what you are
          doing.  If you want to view the parameters, you can
          answer "n" to this question and exit the form without
          changing them.

              Do you want to accept the default disk parameters? _
```

Answer such questions by typing y or n and pressing [ENTER].

Other forms contain one or more fields for you to fill in. If a value appears in the form when it is first displayed, that value is either a recommended default value or the operating system has determined that it is the correct value for your computer. For example, the `DISK PREPARATION` form displays with default values already on the form:



```
Install  Update   Help     Shell    Exit
Do full installation

              DISK PREPARATION: BOOT DISK

                Format disk?            N

                Partition disk?         N

                Interleave factor:      0

                Surface analysis?     < read >

                Specify disk defects?   Y

              ESC to exit, F1 for help
```

Move *between* fields using ENTER or the TAB and BACK-TAB keys. The cursor wraps from the bottom item back up to the top and vice versa.

Some fields are present for your information only and cannot be changed. The system does not allow you to move the cursor to a field that cannot be changed. If you are using a color monitor, fields that can be changed are highlighted in blue. The currently active field appears in reverse video on monochrome monitors and is highlighted in white on color monitors.

Move *within* a field using the left and right arrow keys. Certain fields allow you to toggle through your choices using the spacebar. These fields are surrounded by the symbols < and > (like the `Surface analysis?` field above). Otherwise, enter data into the fields by typing it in. The default mode is to overwrite text, but you may toggle between the overwrite and insert modes using the INSERT key. The BACKSPACE and DELETE keys work as you would normally expect.

If the system "beeps" at you and does not respond, you have entered a value that is not allowed or have attempted to use an inappropriate key. Delete your entry and try again or consult your Quick Reference Card for the valid keys.

As you finish with each field, press ENTER. The system then checks the validity of your answer and displays an error message at the bottom of your screen if there is a problem. If a problem occurs, you will not be allowed to leave the field until you have entered an acceptable value. Some fields may not be left blank.

When you are finished entering data, press ESC. The system then displays this message at the bottom of your screen:

```
Press Y to confirm, N to cancel, E to continue editing
```

This gives you a chance to confirm your input (`Y`), return the form to its default values and re-edit it (`N`), or leave the values as they are and continue editing (`E`). You do not need to press ENTER after making this choice; the system acts as soon as `y`, `n`, or `e` is pressed.

Press F1 to display help on the current field. If no context-specific help is available, the Help Index appears instead. If context-specific help is available, pressing F1 a second time displays the Help Index.

4. INSTALLING THE INTERACTIVE UNIX OPERATING SYSTEM

This section outlines the full installation and update procedures in greater detail than is possible on-line. Users who are already familiar with the INTERACTIVE UNIX System do not need to read this section in its entirety.

As discussed in the previous section, the installation procedure is designed so that you can obtain on-line all the information you need to perform an uncomplicated installation. You can safely explore the on-line system; no destructive actions occur unless you are warned and then confirm that the action should take place. The installation can be terminated by pressing CTRL \, and then typing `shutdown`.

Note that the default state of the numeric keypad has been changed. During the boot, the NUM-LOCK key is turned on, enabling the numeric keypad. If you would like to use the numeric keypad for its cursor motion, page-up/page-down, home and end functions, press the NUM-LOCK key after booting. This step is only necessary during the installation procedure and only if you are using an 84-key keyboard or a portable. On 101-key keyboards, the cursor-movement, page-up/page-down, home, and end keys that are *not* located on the numeric keypad function as expected, regardless of the state of the NUM-LOCK key.

4.1 Deciding Whether to Perform a Full Installation or an Update Installation

If you have never installed the INTERACTIVE UNIX Operating System on your computer, you *must* perform a full installation. This destroys the data on your fixed disk *in the partition on which you install the INTERACTIVE UNIX Operating System.*

If you already have an existing release of the INTERACTIVE UNIX System on your computer, you may be able to perform an update rather than a full installation. An update is a nondestructive procedure that makes the minimum number of changes necessary (such as replacing certain system files), but it does not disturb any data. If the new release allows updating and the on-line procedure finds an existing INTERACTIVE UNIX Operating System when it examines your fixed disk, you can either update or perform a full installation.

☛ Regardless of the type of installation you perform, *be sure* to read the release notes that accompanied your software *before* you begin the installation.

Section 4.3 presents a step-by-step example of the full installation procedure. Read your release notes to find out how to perform an update installation.

4.2 Booting the System

The INTERACTIVE UNIX Operating System is delivered on a set of 5¼ inch or 3½ inch diskettes that you install onto your fixed disk.

☛ Before you begin installation, be sure you have the serial number and authorization key that accompanied your software.

When you install a system, the first task is to *boot* (bring up) the system. Take out the two system diskettes labeled *Boot* and *Install* and follow these steps to start the system:

1. Insert the diskette labeled *Boot* into the high density diskette drive 0.

2. Turn on any expansion units (such as an external fixed disk) that are present.

3. Turn the computer on. (If the power is already on, turn the computer OFF, then turn it ON again.)

   ☛ On some systems, you can simultaneously press $\boxed{\text{CTRL}}$, $\boxed{\text{ALT}}$, and $\boxed{\text{DEL}}$ to reboot a system that is already installed on a fixed disk or to reboot a computer system when the power is already on. On other systems, there is a RESET switch or button that can be used. You can always turn the power off then on again to restart the installation process on any system.

   Your screen will look similar to this:

   ```
   Booting the INTERACTIVE UNIX Operating System
   ```

4. When the system has been booted from the *Boot* diskette (note that this may take a while), some system-dependent messages are displayed, followed by the software copyright notices. The system then displays this message:

   ```
   Insert the Install diskette and press <ENTER>
   ```

5. Note that if you press [ENTER] when there is no diskette in the drive or after inserting the wrong diskette, the system will lock up. You will have to power down the machine, insert the *Boot* diskette, and begin the installation over again.

   Remove the *Boot* diskette and insert the diskette labelled *Install* into the drive and press [ENTER]. More copyright notices are displayed while the installation software is loaded.

6. You are now prompted to enter the serial numbers and authorization key for your INTERACTIVE UNIX System software. Type in this information from the card included with your software.

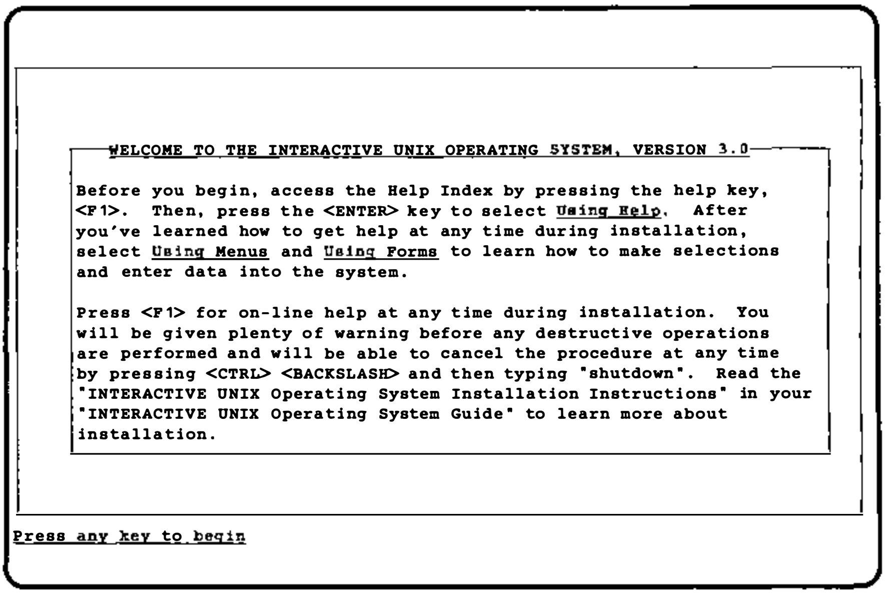
   ☛ If you do not have the serial number and/or authorization key, you will have to find the missing information before you can continue. At this point, it is impossible to abort the installation cleanly. It is recommended that you *do not* power off the computer until you can continue the installation unless it is absolutely unavoidable.

   If your authorization code is valid, the system confirms this and prompts you to select the type of national keyboard you are using.

7. Use the up and down arrow keys to select the type of keyboard you have and press [ENTER]. The screen then refreshes and the system asks:

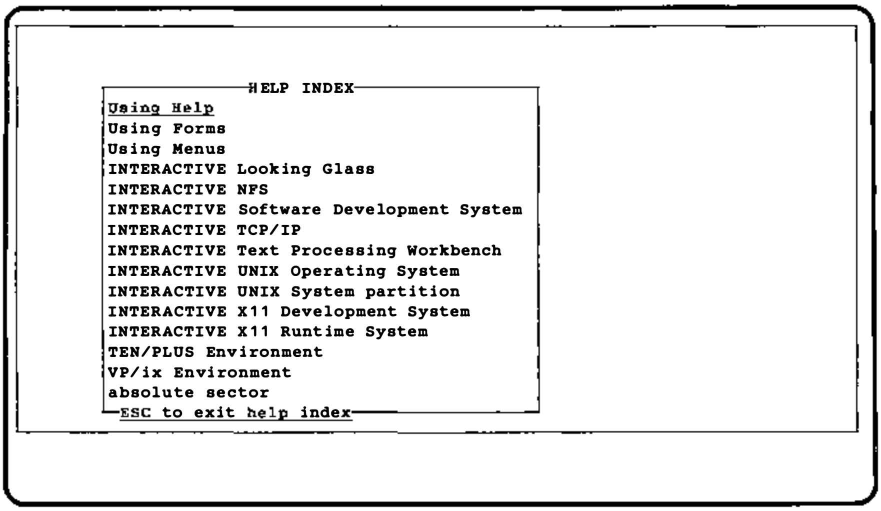
   ```
   Do the lines in this box appear in different colors?
   ```

8. If you are using a color monitor and the display on your screen appears in color, type y and press [ENTER]. If you have a monochrome monitor, the display will not appear in color; type n and press [ENTER]. The system then displays an information screen:



```
WELCOME TO THE INTERACTIVE UNIX OPERATING SYSTEM, VERSION 3.0

Before you begin, access the Help Index by pressing the help key,
<F1>.  Then, press the <ENTER> key to select Using Help.  After
you've learned how to get help at any time during installation,
select Using Menus and Using Forms to learn how to make selections
and enter data into the system.

Press <F1> for on-line help at any time during installation.  You
will be given plenty of warning before any destructive operations
are performed and will be able to cancel the procedure at any time
by pressing <CTRL> <BACKSLASH> and then typing "shutdown".  Read the
"INTERACTIVE UNIX Operating System Installation Instructions" in your
"INTERACTIVE UNIX Operating System Guide" to learn more about
installation.

Press any key to begin
```

9. If you have read about using the on-line system in the previous section, you do not need to read the Help Index topics and can press any key to continue with the installation. You should then skip to step 11. If you have *not* read the previous section, you should access the Help Index by pressing the help key, $\boxed{\text{F1}}$. When you access the Help Index, the cursor will always be on the topic `Using Help`.



```
HELP INDEX
Using Help
Using Forms
Using Menus
INTERACTIVE Looking Glass
INTERACTIVE NFS
INTERACTIVE Software Development System
INTERACTIVE TCP/IP
INTERACTIVE Text Processing Workbench
INTERACTIVE UNIX Operating System
INTERACTIVE UNIX System partition
INTERACTIVE X11 Development System
INTERACTIVE X11 Runtime System
TEN/PLUS Environment
VP/ix Environment
absolute sector
ESC to exit help index
```

10. Press ENTER to select `Using Help` and read the information there. Press any key to exit that topic. Select and read `Using Forms` and `Using Menus`, as well. Then press ESC to exit from the Help Index and continue with the installation.

11. The system presents a list of INTERACTIVE UNIX System software packages and asks you to indicate the package you want to install. Use the up and down arrow keys to highlight the package you are installing and press ENTER.

12. The system then asks you whether you want additional help information to be displayed throughout the installation procedure. If you are a new INTERACTIVE UNIX System user, answer `y` to this question. If you are an experienced INTERACTIVE UNIX System user and you are familiar with the INTERACTIVE UNIX System installation procedure, you may want to answer `n` to suppress the information screens.

13. The system now checks for an existing INTERACTIVE UNIX Operating System on your fixed disk.

    ☛ At this point and others during installation, depending on what type of fixed disk you have, your fixed disk may spin down and then spin back up again. This is normal and will not impede your installation.

If the system does *not* find an existing INTERACTIVE UNIX Operating System or if an update is not possible, this information is displayed on your screen. You must then perform a full installation. In this case, press any key to continue the installation and skip to the next section, "A Step-by-Step Example of a Full Installation."

☛ In order to install or update the INTERACTIVE UNIX Operating System, your fixed disk controller or SCSI host adapter must be set up in the configuration expected by the installation kernel on the *Boot* diskette. If it is not, your system may hang at this point because it is the first time an attempt is made to access the fixed disk during installation. Normally, this should not happen, since the installation kernel is configured to match the factory settings of all disk controllers and SCSI adapters. However, if your system does hang, consult the list of controllers in section 8.2.2 of the "INTERACTIVE UNIX Operating System Maintenance Procedures" and determine which *Controller Module* corresponds to your primary fixed disk controller. Then consult the tables in section 8.2.3 to determine which, if any, compatibility mode your controller should be in and which I/O address, IRQ level, and DMA channel it should be using. You must ensure that your controller is set to these values by checking your hardware manufacturer's documentation, and if necessary rejumpering your controller or running a configuration utility.

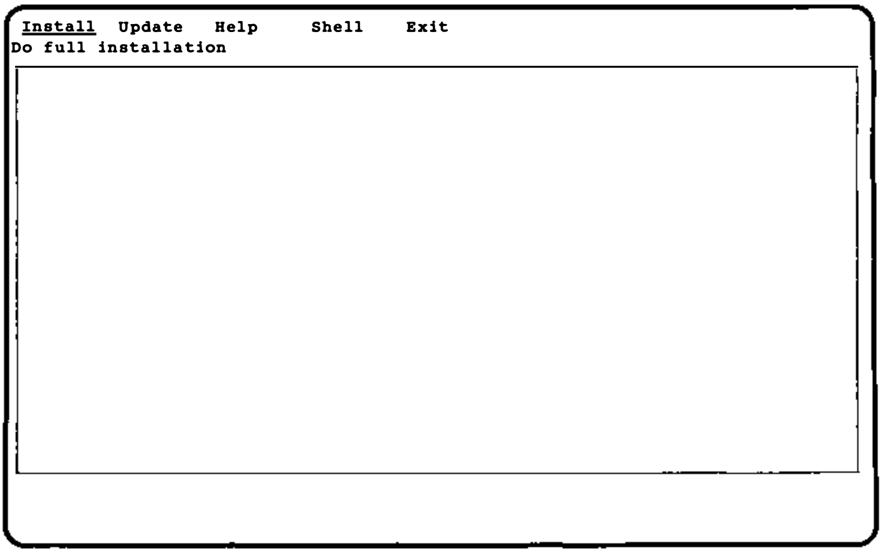
Note that once the INTERACTIVE UNIX Operating System has been installed, it is suggested that you reconfigure the kernel (using `kconfig`) to support any controller configuration. In particular, if you have installed your controller in a compatibility mode, you will want to configure your kernel to run it in its native mode to take advantage of all of the controller's features. For more information on kernel configuration, refer to section 8.2 of the "INTERACTIVE UNIX Operating System Maintenance Procedures."

If the INTERACTIVE UNIX System *does* find an existing INTERACTIVE UNIX System and an update is possible, a different information screen is displayed. You may perform either an update or a full installation. If you plan to perform a full installation, press any key to continue and proceed to the next section.

If you plan to perform an update, you should refer to your release notes for any special information about updating the system. Then press any key to continue, select the `Update` option from the next menu, and skip to section 4.5, "Installing the Base Operating System."

4.3 A Step-by-Step Example of a Full Installation

At this point in the installation procedure, the installation menu is displayed:



```
 Install  Update  Help     Shell    Exit
Do full installation
```

Note that if it is not possible for you to perform an update, the `Update` option will not appear on your screen.

1. To perform a full installation, select `Install` and press ENTER. The system takes a few seconds to determine the parameters (characteristics) of your fixed disk and then asks you whether you want to accept the parameters it has determined.

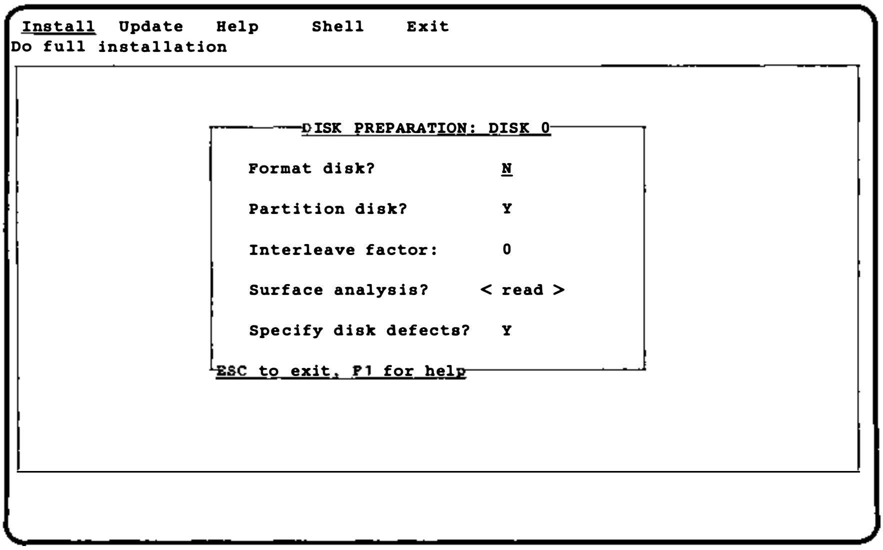
2. Type `y` to accept the default disk parameters. In most cases, the system functions smoothly if you accept the defaults. Do *not* attempt to change the parameters unless you understand

what they represent and are sure you know what you are doing.

However, if you are performing a complete (destructive) installation of the INTERACTIVE UNIX System onto a fixed disk with more than 1024 cylinders, you will want to answer `n`. Enter the correct number of cylinders available to the INTERACTIVE UNIX System into the form. If your system has more than 1024 cylinders and some are reserved for diagnostics or defect management, enter the number that the INTERACTIVE UNIX System can use. (This applies when you are adding additional disks using `sysadm addhd` as well.)

4.3.1 Questions for Disk Preparation

The system now asks some questions about your fixed disk. It displays the following:



```
 Install  Update  Help    Shell   Exit
Do full installation

                  DISK PREPARATION: DISK 0

                 Format disk?            N

                 Partition disk?         Y

                 Interleave factor:      0

                 Surface analysis?     < read >

                 Specify disk defects?   Y

                ESC to exit, F1 for help
```

The form is displayed with the recommended default values in place. You can use the spacebar to toggle through the acceptable values for each field (except `Interleave factor:` for which you must enter a number).

Remember that, as with all forms used in the INTERACTIVE UNIX System installation procedure, it is quite safe for you to explore the form if you are unsure about how to use it or what values to enter. No action will be taken until you explicitly confirm that you have finished editing. Use the $\boxed{\text{TAB}}$ and $\boxed{\text{BACK-TAB}}$ keys to move between fields, and experiment with entering values. Use the $\boxed{\text{F1}}$ key to display context-specific help on the currently highlighted field.

4.3.1.1 Formatting the Fixed Disk. On most systems, it is required that you format (or reformat) your fixed disk when you install a new system. In general, it is recommended that you use the format disk utility supplied by the disk manufacturer. This means that you should skip (or type `no` to) the `format` option during the INTERACTIVE UNIX Operating System installation procedure. Use this option only as a last resort if your first attempt at installation fails.

Note that some disk manufacturers specify that their disks must never be reformatted outside the factory or without unusual procedures. For example, COMPAQ 386* fixed disks are formatted at the factory and should *not* be formatted when installing the INTERACTIVE UNIX Operating System.

If you already have another operating system partition that you want to keep on your disk, you should not reformat the disk until you have backed up that partition so that you can reinstall it after the INTERACTIVE UNIX Operating System is installed.

4.3.1.2 Partitioning the Fixed Disk. If you have an unused fixed disk or you have just reformatted your disk, it will not contain any partitions. You must partition it, even if you plan to install the INTERACTIVE UNIX Operating System on the entire disk. You can use the entire disk for the INTERACTIVE UNIX Operating System or you can establish a second partition (and optionally, third and fourth partitions) for MS-DOS or another operating system, such as XENIX or OS/2. (Remember that you can run XENIX applications on the INTERACTIVE UNIX System without running the XENIX operating system on a separate partition and that if you have the VP/ix Environment, you do not need a separate partition for the MS-DOS operating system to run MS-DOS applications.)

If you *do* plan to use more than one operating system on your machine, you must divide your fixed disk into at least two partitions. When establishing more than one partition, begin the non-INTERACTIVE UNIX System partition on cylinder 0, and begin the INTERACTIVE UNIX System partition on the cylinders above the first partition. The INTERACTIVE UNIX System partition must be the *active* partition.

4.3.1.3 Your Fixed Disk Interleave Factor. The interleave factor is used when a disk is formatted to determine how the sectors within a track should be numbered. On some disks, sectors within a track are not numbered and read sequentially. How they are numbered depends on the rate at which the computer can most efficiently read the data as the sectors pass under the recording head. Fixed disks function most efficiently if an appropriate interleave factor is used when reading them.

If you use the INTERACTIVE UNIX System to format your disk, you must enter an interleave value. Even if you have decided not to format your disk, knowing the appropriate interleave factor helps the system to mark bad tracks in the most space-efficient manner. If you do not format your disk and do not know the appropriate interleave factor, it is safe to enter 0.

If the system displays a value other than 0 in this field, it was automatically determined from your disk. *Do not change* this value unless you are formatting and are sure that a different value is more suitable. Refer to the manufacturer's information supplied with your disk to determine the appropriate interleave factor. Interleave factors are frequently reported as a ratio, for example, 3:1. In this case, the interleave factor is 3. Values ranging from 0 to 9 are acceptable (unless you choose to format your disk, in which case the value must not be 0).

4.3.1.4 Performing a Surface Analysis. You may perform a complete (write) surface analysis, a partial (read) surface analysis, or none at all. We recommend that you perform a surface analysis unless you have a controller or disk that is capable of compensating for its defects and hiding them from the system, for example, a SCSI disk. It is a good idea to perform a complete surface analysis if you have never before installed the INTERACTIVE UNIX Operating System on your machine.

A complete analysis reads and writes every portion of the INTERACTIVE UNIX System partition. It may take quite a long time, depending on your system.

☛ Note that performing a complete surface analysis is time-consuming and *destroys all data on the INTERACTIVE UNIX System partition.* However, it usually finds more errors.

The partial analysis is faster and is also nondestructive; choose this option if you have data in an existing INTERACTIVE UNIX System partition that you want to preserve. Note that if you are performing a full installation, all data on your disk will be destroyed.

4.3.1.5 Specifying Known Bad Sectors on the Disk. The manufacturer usually supplies a list of defects with each fixed disk. The defects are either listed in the documentation supplied by the manufacturer or printed on a label attached to the fixed disk itself. Defects are potentially unusable portions of the disk that are discovered during the manufacturer's testing.

If you have a disk or controller that is capable of compensating for its defects without involving the operating system (for example, a SCSI disk), your manufacturer may not supply a defect list. If your drive manufacturer does supply a defect list, answer y to this question so that you can enter them into the defect table on the disk. The system can then avoid writing data into these bad spots, guarding against data loss.

You do not need to enter the defects on the manufacturer's list if you are using a SCSI disk or ESDI drive on the IBM* PS/2* model 80, because any sectors in the manufacturer's defect list will automatically be redirected to alternates by the controller during formatting.

4.3.1.6 Exiting the Disk Preparation Form.

1. When you have finished with the DISK PREPARATION form, press $\boxed{\text{ESC}}$. The system displays this message at the bottom of your screen:

```
Press Y to confirm, N to cancel, E to continue editing
```

2. Press y to confirm your input, n to return the form to its default values and re-edit it, or e to leave the values as they are and continue editing. (Note that you do not need to press $\boxed{\text{ENTER}}$ after making this choice. The system acts as soon as y, n, or e is pressed.)

If you chose to format the disk, to run some form of surface analysis, and/or to partition the disk, they will be performed when you exit from this form. If you did not choose to partition the disk, skip the following section.

4.3.2 Formatting Your Disk

If you answered `n` to the `Format disk?` question, skip to the next section.

The system will give you one more opportunity to change your mind about formatting the disk (which will destroy all data stored on it). You should be sure that you have a disk that can be reformatted by the operating system. If you already have another operating system partition on your disk that you want to keep, be sure you have backed up that partition.

If you confirm that you want to format the disk, the formatting process then begins. Progress messages are displayed at the bottom of the screen to allow you to keep track of the formatting and to estimate how long it will take.

4.3.3 Running the `fdisk` Program to Partition the Disk

If you answered `n` to the `Partition disk?` question, skip to the next section.

If you chose to partition your fixed disk, the system prompts you to press any key to run the `fdisk` program.

1. Press any key. The `fdisk` program, which is responsible for the display on your screen, prompts you through the procedure. If you have ever used the `fdisk` program on your fixed disk, skip to step 3. If you have *never* used the `fdisk` program on your fixed disk before, your screen will look similar to this:

   ```
   Do you want to partition your hard disk as follows:

   85% "UNIX" - lets you run UNIX programs
   15% "DOS" - lets you run DOS without UNIX

   To do this, please type "y".  To partition your hard disk
   differently, type "n" and the fdisk program will let you
   select other partitions.
   ```

2. Type `y` if this division is acceptable to you and you do not want to establish a partition for any additional operating system. Type `n` if you want to choose different sizes or establish more than two partitions.

3. Your screen will look similar to this, depending on the size of your fixed disk and the number and type of partitions already on it:

```
Available hard disk size is 823 cylinders.

                                               Cylinders
  Partition  Status  Type           Start  End  Length    %
  ---------  ------  --------       -----  ---  ------   ---
      1      Active   DOS               0  822     823   100
      2               UNUSED
      3               UNUSED
      4               UNUSED


SELECT ONE OF THE FOLLOWING

   1. Create a partition
   2. Change Active (Boot from) partition
   3. Delete a partition
   4. Display Partition Table
   5. Exit (Update disk configuration and exit)
   6. Cancel (Exit without updating disk configuration)
Enter selection:
```

If you have an existing `DOS` or `Other` type of partition that is small enough to allow creation of an INTERACTIVE UNIX System partition, you do not need to delete the existing partition. If you have a `DOS` or `Other` type partition that uses most or all of the fixed disk, you must delete it before continuing.

☛ Note that *deleting a partition destroys all files in that partition*. Before you delete it, be sure you have backed up any files you want to save. (For more information on backing up files, refer to section 12, "BACKING UP FILES," in "System Administration for New Users of the INTERACTIVE UNIX Operating System.")

If your disk is large enough to have several DOS partitions (one primary and up to 11 extended partitions) and you want to keep them all, you will be able to display information about them using option 4 above, unless you see the following message:

```
Do you want to fix the Extended DOS partitions for UNIX
access?
```

4. If this message appears on your system, you must answer `y` in order to display information about them using option 4 above.

5. To delete the partition, type 3 and press ENTER, then type the number of the partition you want to delete. Your screen will look similar to this:

```
Available hard disk size is 823 cylinders.

                                        Cylinders
Partition  Status  Type            Start  End  Length   %
---------  ------  --------        -----  ---  ------  ---

 There are no partitions currently defined



SELECT ONE OF THE FOLLOWING

   1. Create a partition
   2. Change Active (Boot from) partition
   3. Delete a partition
   4. Display Partition Table
   5. Exit (Update disk configuration and exit)
   6. Cancel (Exit without updating disk configuration)

Enter selection:
```

6. Now create a partition by typing 1. Your screen will look similar to this:

```
Indicate the type of partition you want to create
(1=UNIX System, 2=MS-DOS only, 3=Other, x=Exit).
```

7. Type 1 to create the INTERACTIVE UNIX System partition. The system displays:

```
The UNIX System partition must use at least nnn% of the
hard disk.  Indicate the percentage (nnn-100) of
the hard disk you want this partition to use
(or enter "c" to specify in cylinders).
```

8. If you plan to have only one partition, type 100. If you plan to have more than one partition, type a number that is at least as large as the lower percentage that appears on the previous screen.

   Note that if your disk has more than 1024 cylinders and you make the INTERACTIVE UNIX System partition larger than that, the following message appears:

```
Ending cylinder n truncated to 1023 due to ROM BIOS limits
on fdisk table.
But do not worry, UNIX can use all n cylinders of the disk.
```

   The ending cylinder number appears smaller than it actually is, but the INTERACTIVE UNIX System will use all cylinders and access the disk correctly.

   Your screen will then look similar to this:

```
Do you want this to become the Active partition?
If so, it will be activated each time you reset
your computer or when you turn it on again.
Please type "y" or "n".
```

9. Type y to make the INTERACTIVE UNIX System partition your active partition. Only one partition can be active at a time; this must be your INTERACTIVE UNIX System partition. (Note that if you have an extended DOS partition, of type `EXTDOS`, it may never be made active.) The system displays:

   ```
   Partition 1 is now the Active partition.
   ```

   After the partition is created, your screen will look similar to this (if you plan to have only one partition):

   ```
   Available hard disk size is 823 cylinders.


                                           Cylinders
   Partition  Status  Type             Start  End  Length   %
   ---------  ------  ---------        -----  ---  ------  ---
       1      Active   UNIX System        0   822     823  100
       2               UNUSED
       3               UNUSED
       4               UNUSED

   SELECT ONE OF THE FOLLOWING

       1. Create a partition
       2. Change Active (Boot from) partition
       3. Delete a partition
       4. Display Partition Table
       5. Exit (Update disk configuration and exit)
       6. Cancel (Exit without updating disk configuration)

   Enter selection:
   ```

10. If you want to create additional partitions to contain other operating systems, select 1 again and step through this same procedure. If you attempt to create a DOS partition that is larger than the 32 MB limit imposed by DOS, the following message appears:

    ```
    You have exceeded the maximum number of DOS sectors
    allowed in a DOS partition.  Maximum cylinder size will be n

    The partition will be created with the maximum cylinder
    size allowed.
    ```

    The DOS partition is created with the maximum cylinder size allowed.

11. When you have finished creating partitions, type 5 to exit the program. If you exit the `fdisk` program without making the

INTERACTIVE UNIX System partition the *active partition* or if the system is unable to locate an INTERACTIVE UNIX System partition, you are automatically returned to `fdisk` until you correct the problem.

Note that if the *first* cylinder in the INTERACTIVE UNIX System partition contains bad sectors, the installation will fail later during the procedure. In this case, the system displays an error message that gives the cylinder number you should use as the first cylinder of the INTERACTIVE UNIX System partition. Restart the installation and when `fdisk` is run, use the value you are given by the system as the beginning of the INTERACTIVE UNIX System partition.

4.3.4 Running a Surface Analysis on the Disk

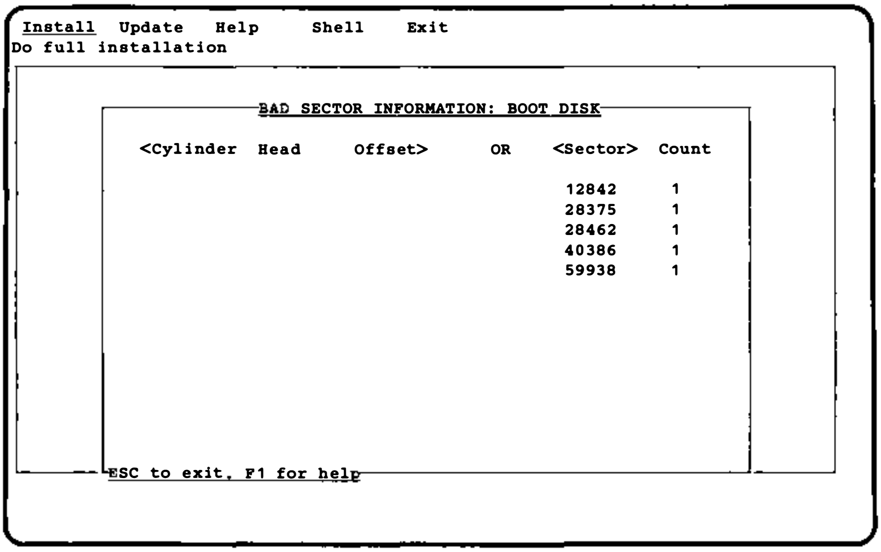
If you answered `n` to the `Surface analysis?` question, skip to the next section.

If you chose a partial analysis, the analysis proceeds at this point. If you chose a complete, destructive analysis (`write`), the system gives you an opportunity to change your mind and choose to partial (`read`) instead.

Progress messages are displayed at the bottom of the screen to allow you to keep track of the analysis and to estimate how long it will take. A surface analysis can be quite time-consuming, depending on the size of your disk.

4.3.5 Entering Known Bad Sectors on the Fixed Disk

If you did not choose to specify your disk defects, skip this section. If you chose to specify your disk defects, the system next displays an information screen followed by the `BAD SECTOR INFORMATION` form:



```
 Install  Update  Help    Shell   Exit
Do full installation

                 BAD SECTOR INFORMATION: BOOT DISK

      <Cylinder  Head     Offset>     OR    <Sector>  Count

                                             12842     1
                                             28375     1
                                             28462     1
                                             40386     1
                                             59938     1

  ESC to exit, F1 for help
```

If you ran a surface analysis, the bad sectors found by the system are already entered into the form in *absolute sector* format (the `<Sector>` column). If the system finds more errors than the form can accommodate, they are not displayed. Instead, a message appears to explain that these bad sectors can be viewed in the `/etc/partitions` file after installation is complete.

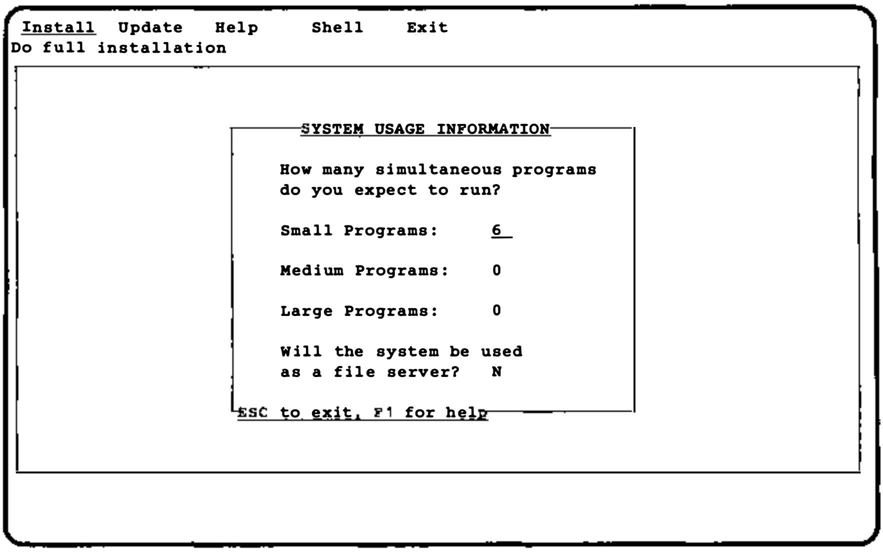
Your manufacturer's defect list will probably be in *cylinder*, *head*, and *offset* format. As you enter new sectors in this format, the system automatically converts these entries to absolute sector values. If you have bad sector information previously reported by the INTERACTIVE UNIX System, enter it directly in the absolute sector format. If a defect is difficult to locate or falls very near a sector boundary, it may require several sectors to be marked as bad. This information appears in the `Count` field as you enter the defects.

Use the arrow keys to move up and down as necessary. The screen scrolls automatically to make room for all entries. Use the **TAB** and **BACK-TAB** keys or the **ENTER** key to move from field to field.

To delete an entry you have made in cylinder, head, offset format, erase the cylinder field and type in 0; to delete an absolute sector format entry, type in 0 in the sector field. When you have finished entering the bad sector data, press **ESC** to exit the form.

4.4 Dividing Your INTERACTIVE UNIX System Partition

After you exit the BAD SECTOR INFORMATION form, the system displays an information screen followed by the SYSTEM USAGE INFORMATION form:



```
SYSTEM USAGE INFORMATION

How many simultaneous programs
do you expect to run?

Small Programs:      6

Medium Programs:     0

Large Programs:      0

Will the system be used
as a file server?    N

ESC to exit, F1 for help
```

The information you provide on this form is used to calculate a reasonable default value for the INTERACTIVE UNIX System swap partition. (You will have an opportunity to change the default value later.) The default values for system usage already appear on the screen.

1. In the first field, enter the number of small programs you expect to run simultaneously on your system. A small program is defined as one that uses 2 MB or less of system memory. Examples are editors, compilers, and applications that do not use graphics. If you are unsure of how much memory any of your applications use, refer to the documentation supplied with each application or ask the manufacturer.

2. In the second field, enter the number of medium-sized programs you expect to run simultaneously on your system. A medium program is defined as one that uses 2 to 3 MB of system memory. Examples are the VP/ix Environment and most graphics applications written for non-windowing environments.

3. In the third field, enter the number of large programs you expect to run simultaneously on your system. A large program is defined as one that uses 3 or more MB of system memory. Examples of large programs are graphics applications written for windowing environments, such as a CAD/CAM package.

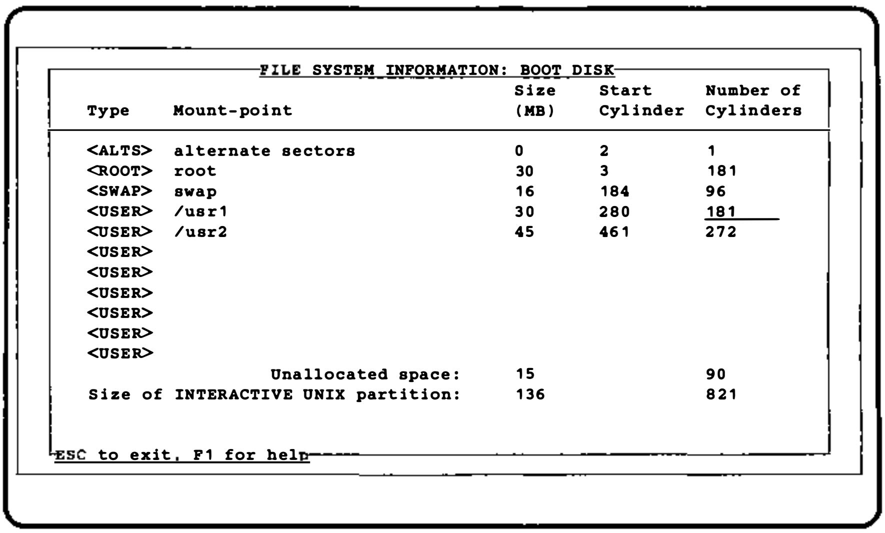
4. The last field asks if you plan to use your computer as a *file server*. If you are going to use your computer primarily to contain shared file systems in a network environment, such as INTERACTIVE TCP/IP or INTERACTIVE NFS*, you need a larger swap space than a computer that will not be used in a network. This information is used to increase the amount of swap space allotted by the INTERACTIVE UNIX System. (Swap space or area is discussed in the next section of this document.)

5. When you are satisfied with the values on this form, press $\boxed{\text{ESC}}$, then confirm your changes by pressing `y`.

4.4.1 Creating INTERACTIVE UNIX System File Systems

Based on the information you just provided, the system next calculates the recommended amount of space on your fixed disk for swap space and the INTERACTIVE UNIX System file systems, such as `root` and `usr`.

1. The system displays some information screens. Press any key to display the `FILE SYSTEM INFORMATION` form. It looks similar to this:



```
                    FILE SYSTEM INFORMATION: BOOT DISK
                                                  Size    Start     Number of
  Type     Mount-point                            (MB)    Cylinder  Cylinders

  <ALTS>   alternate sectors                      0       2         1
  <ROOT>   root                                   30      3         181
  <SWAP>   swap                                   16      184       96
  <USER>   /usr1                                  30      280       181
  <USER>   /usr2                                  45      461       272
  <USER>
  <USER>
  <USER>
  <USER>
  <USER>
  <USER>
                     Unallocated space:           15                90
  Size of INTERACTIVE UNIX partition:             136               821

ESC to exit, F1 for help
```

This form allows you to specify the number and size of the file systems you want to create on your fixed disk. The form operates like a spreadsheet. As you enter a value into one field, the values of the other fields are recalculated in order to ensure that the totals and the relationships among the fields are always correct.

Some systems may contain more than one fixed disk. This form is used to divide your primary, or *boot*, fixed disk. This is the disk from which the system will be started each time you shut the system down and reboot.

4.4.1.1 The `alts` *Area.* The `alts` area holds alternate sectors used to compensate for bad sectors on your fixed disk. You cannot decrease its size.

4.4.1.2 The `swap` *Space.* Because the INTERACTIVE UNIX Operating System is a multi-user, multi-tasking system with many processes running simultaneously, a `swap` space is needed. The `swap` space is the area where partial or complete processes (programs in execution) are temporarily transferred from memory to wait for main memory to become available again.

Generally speaking, the more users and/or less memory your system has, the larger the amount of `swap` space needed. If you are using networking packages such as INTERACTIVE NFS or INTERACTIVE TCP/IP, you will need to increase your `swap` space.

4.4.1.3 The `root` *File System.* The `root` file system holds the files needed for operating system functions and many of the applications and networking packages you use. If you know which INTERACTIVE UNIX System subsets and extensions you plan to install, you can look at the release notes that accompanied your software to see approximately how much memory each package needs. Add to this the memory needed for any other software packages you plan to install and you can calculate a reasonable value for your `root` file system.

4.4.1.4 The `tmp` *File System.* The `tmp` file system holds temporary files created by a number of system processes; these files are deleted each time the system is shut down. Note that it is not necessary to make a separate file system to hold temporary files. The `/tmp` directory in the `root` file system is used for this purpose if no `tmp` file system is created. Also, it is often difficult to estimate the correct size for the `tmp` file system. If it is too small, some programs may fail during execution; if it is too large, the extra space is wasted. Unless you are confident that you can estimate the required size, you should not specify a `tmp` file system.

4.4.1.5 The User File Systems. The first user file system (`usr`) holds some system files and, optionally, files created by users. The other user file systems (up to eight user file systems may be created) hold files created by users. (Note that you do *not* need to have a `usr` file system for each system user.)

4.4.1.6 Changing the Size of File Systems. Change the `Size` field of each file system to the number of megabytes you want to allocate. As you change the size of each file system, the installation program automatically recalculates the starting cylinder number of each file system and adjusts the amount of space in the `Unallocated space:` field to maintain the correct totals. You may specify file system sizes in cylinders, but you will probably find it simpler to use megabytes.

When you make a file system smaller (or delete it), its space is allocated to the pool of unallocated space. If you make a file system bigger or add a new file system, the necessary space is taken from the pool. Before you can increase the size of a file system, therefore, you must delete or decrease the size of another file system in order to free up the necessary space.

Note that sometimes the totals of all allocated and unallocated space do not exactly match the total amount of space available in the INTERACTIVE UNIX System partition. This is due to rounding errors (typically, the number of cylinders per megabyte is not a whole number) and will not affect the operation of the system.

The following restrictions apply to file system sizes:

- You may not decrease the size of the `alts` area. The default values that appear on your screen is the minimum required by the INTERACTIVE UNIX System to provide alternates for the current known bad sectors and those estimated to develop in the future. The system calculates the default based on the bad sectors found during surface analysis and those entered from the manufacturer's defect list. It should be more than sufficient for a typical system.
- You may not decrease the size of the `root` and `usr` file systems and `swap` space below certain minimums required by the INTERACTIVE UNIX System software.

4.4.1.7 Adding and Deleting File Systems. To delete a file system, simply erase the contents of its `Mount-Point`, `Size`, or `Number of Cylinders` field. To add a new file system, ensure that there is sufficient space available in the `Unallocated space:` field, then modify the `Mount-Point` and `Size` fields.

4.4.1.8 Changing the Mount Point. By default, the system provides mount points for the user file systems. It is recommended that these defaults be used, but it is possible to change the names to indicate other devices where the user file systems are to be mounted. Simply type the new name over the default shown in the `Mount-point` field.

4.4.1.9 Finishing the Form.

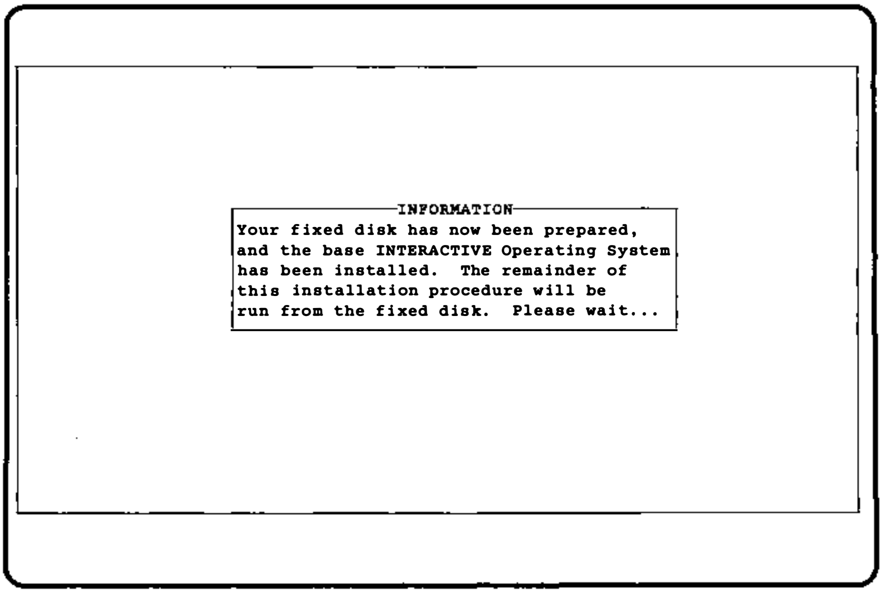
1. When you are satisfied with the values on this form, press $\boxed{\text{ESC}}$. If you have not allocated all of the available space, the system will display an error message; you must allocate all of the available space before you can continue. If you have allocated all of the available space, the system displays:

```
Press Y to confirm, N to cancel, E to continue editing
```

2. Type y when you are ready to exit the form. As soon as you exit, the system will begin to set up your INTERACTIVE UNIX System partition. This can take quite some time.

4.5 Installing the Base Operating System

Regardless of whether you are performing an update or a full installation, the system now copies the new files to the fixed disk. If you are performing a full installation, it also creates directories and installs configuration files at this point. When the system has finished, your screen will look similar to this:



```
┌──────────────────INFORMATION───────────────────┐
│Your fixed disk has now been prepared,          │
│and the base INTERACTIVE Operating System       │
│has been installed.  The remainder of           │
│this installation procedure will be             │
│run from the fixed disk.  Please wait...        │
└────────────────────────────────────────────────┘
```

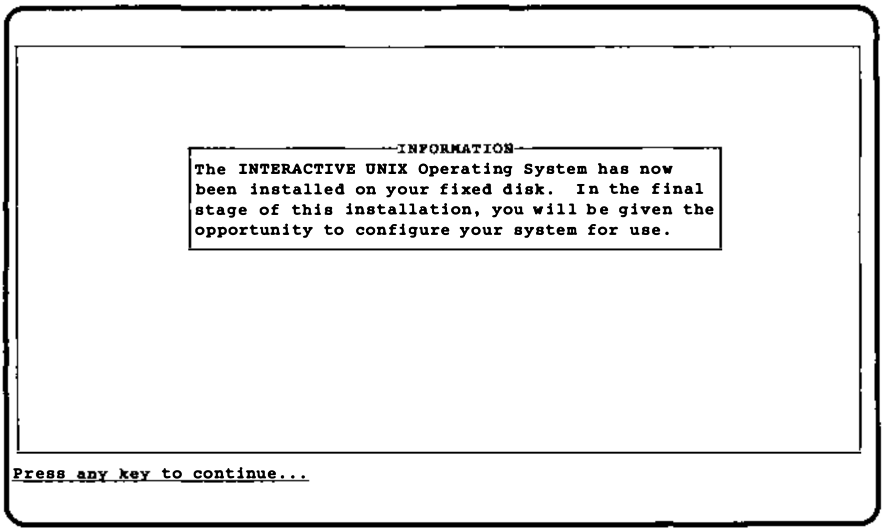
There will be another few minutes' wait while the system finishes the necessary preparations.

The INTERACTIVE UNIX Operating System is delivered with a set of *Core* diskettes, a number of subsets, and any optional extensions you have purchased. You must load the data from the *Core* diskettes onto your fixed disk. The remaining subsets and extensions are optional. Depending upon your individual requirements and the size of your fixed disk, you should install *only* those subsets that are necessary for your planned use of the system. For help in determining which subsets you should install, refer to the release notes that accompanied your INTERACTIVE UNIX Operating System. On some systems, installing *all* the optional subsets can result in poor system performance.

Once the INTERACTIVE UNIX System partition of the fixed disk is completely set up, you are ready to load the Core system onto the fixed disk.

☛ Note that under some circumstances, for example if you have restarted an aborted installation by executing the `Install` script directly from the command line, you may be returned to a shell prompt (#). If this occurs, type `exit` and press ENTER to continue with the installation.

Follow the prompts instructing you to insert the diskettes. As files are copied to your fixed disk, their names are displayed on the screen. When all of the Core system has been installed, a message similar to this is displayed:



```
-----------------------INFORMATION------------------------
The INTERACTIVE UNIX Operating System has now
been installed on your fixed disk.  In the final
stage of this installation, you will be given the
opportunity to configure your system for use.

Press any key to continue...
```

Press any key to continue.

4.6 Initializing the INTERACTIVE UNIX System

If you are performing an update of an existing 386/ix* System or INTERACTIVE UNIX System, you will not be asked to set passwords, the date and time, and the system name. You should now skip to section 4.7.

4.6.1 Setting Passwords for the System and Administrative Logins

The system next gives you the opportunity to establish passwords for the system and administrative logins. An administrative login is used to perform the system administration tasks required to keep the system running smoothly. The system administrator can give just a few privileged users restricted access to perform the tasks that need to be done most frequently. The most important administrative logins are `sysadm`, `powerdown`, `checkfsys`, `makefsys`, `mountfsys`, and `umountfsys`.

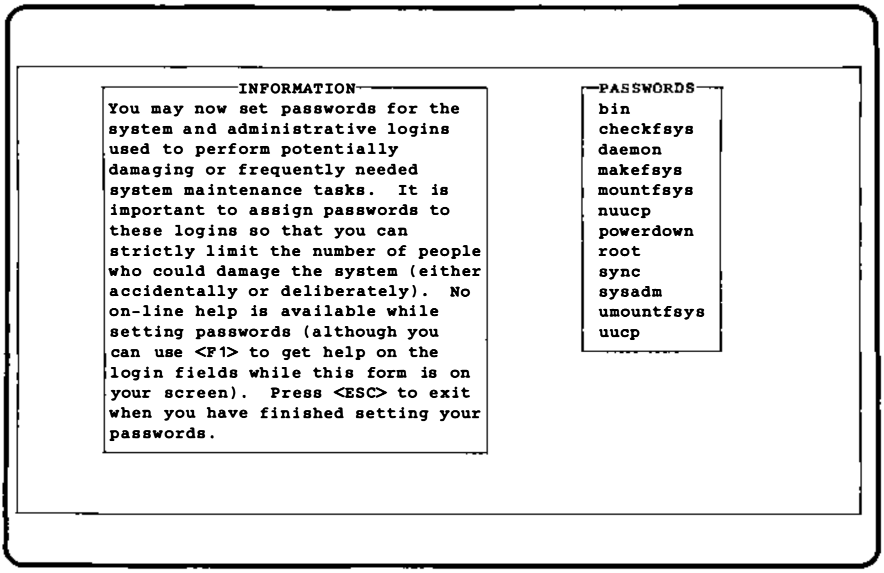
A system login is used to perform system administration tasks that require privileged access to the most restricted files and directories on the system. Some of the system logins on the INTERACTIVE UNIX Operating System include `root`, `bin`, `daemon`, `sync`, `nuucp`, and `uucp`.

Passwords are very important on INTERACTIVE UNIX Systems since they are used to control access to accounts. Passwords prevent unauthorized users from accessing an account and damaging or possibly destroying important data, either accidentally or deliberately. You should set a password for all the important system and administrative accounts. Until you have set passwords for these accounts, it will be impossible to log in to them because they are "locked." The only exception is `root`, which is not locked.

☛ You should set a password for `root` *immediately*, then use the `root` login to set passwords for the other system and administrative accounts.

A password should be a unique word at least six characters long that is not easily guessed. It generally may include upper- and lowercase characters, numbers, and symbols.

No on-line help is available while setting passwords (although you still have access to help while the pop-up menu is on your screen). Your screen will look similar to this:



```
INFORMATION                                PASSWORDS
You may now set passwords for the          bin
system and administrative logins           checkfsys
used to perform potentially                daemon
damaging or frequently needed              makefsys
system maintenance tasks.  It is           mountfsys
important to assign passwords to           nuucp
these logins so that you can               powerdown
strictly limit the number of people        root
who could damage the system (either        sync
accidentally or deliberately).  No         sysadm
on-line help is available while            umountfsys
setting passwords (although you            uucp
can use <F1> to get help on the
login fields while this form is on
your screen).  Press <ESC> to exit
when you have finished setting your
passwords.
```

1. Use the up and down arrow keys to move to the login for which you want to set a password and select it by pressing ENTER. The system displays:

   ```
   New password:
   ```

2. Type in the password. The system displays:

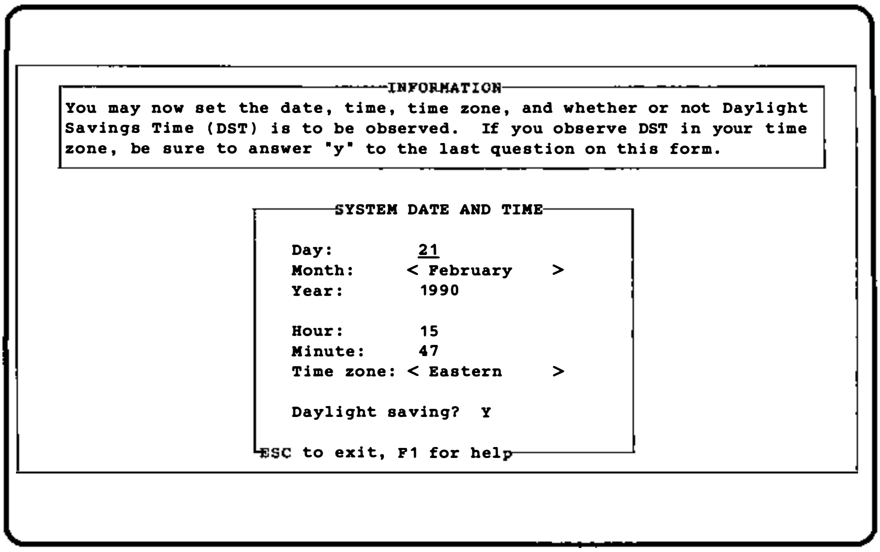
   ```
   Re-enter new password:
   ```

3. Type in the password again. The system returns you to the PASSWORDS menu.

4. When you have finished setting all the passwords you want to set, press ESC to exit.

4.6.2 Setting the Date, Time, and Time Zone

After setting passwords, you have the opportunity to set the system date and time. The system displays:



```
INFORMATION
You may now set the date, time, time zone, and whether or not Daylight
Savings Time (DST) is to be observed.  If you observe DST in your time
zone, be sure to answer "y" to the last question on this form.

            SYSTEM DATE AND TIME

   Day:          21
   Month:      < February     >
   Year:         1990

   Hour:         15
   Minute:       47
   Time zone:  < Eastern      >

   Daylight saving?  Y

ESC to exit, F1 for help
```

Default values are already on the form.

1. Use ENTER or the TAB and BACK-TAB keys to move from field to field, correcting the values where needed. You can use the spacebar to toggle through the acceptable values for the month and time zone.

2. When you are satisfied with the values on this form, press ESC. The system displays:

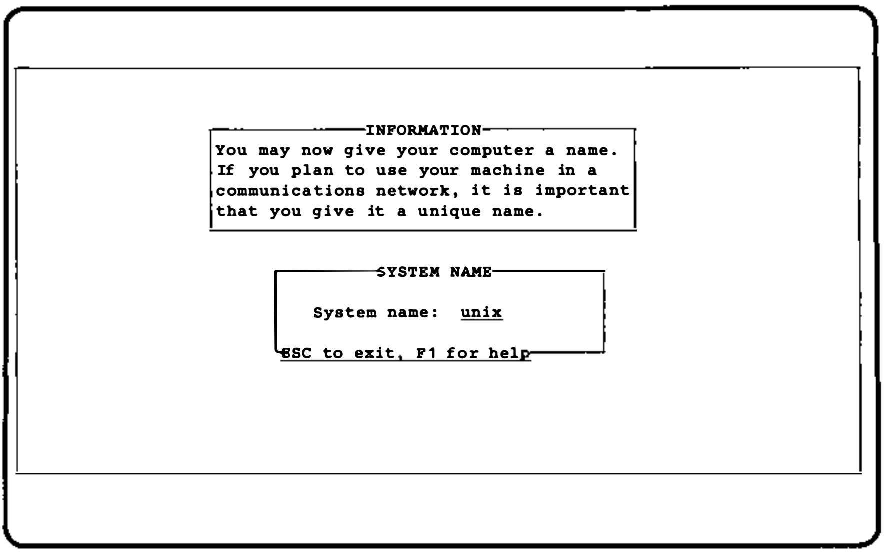
   ```
   Press Y to confirm, N to cancel, E to continue editing
   ```

3. Type `y` when you are ready to exit the form.

Many INTERACTIVE UNIX Systems are used in areas that observe Daylight Saving Time. If the computer is running during the change of Daylight Saving Time, the system time will automatically adjust. However, the next time the system is rebooted, it will pick up the time as maintained in CMOS RAM. You will have to *manually reset* the system clock during each seasonal time change. To make the change permanent, you can use your manufacturer's *setup* program, or use the INTERACTIVE UNIX System `date` command or `sysadm datetime`. You can do this at the same time you reset your other clocks in your home and office. For more details, refer to section 8.4 in "System Administration for New Users of the INTERACTIVE UNIX Operating System" or to *timezone*(4).

4.6.3 Giving Your System a Name

After setting the date and related information, you should give your system a name. The system displays:



```
INFORMATION
You may now give your computer a name.
If you plan to use your machine in a
communications network, it is important
that you give it a unique name.

SYSTEM NAME

   System name:  unix

ESC to exit, F1 for help
```

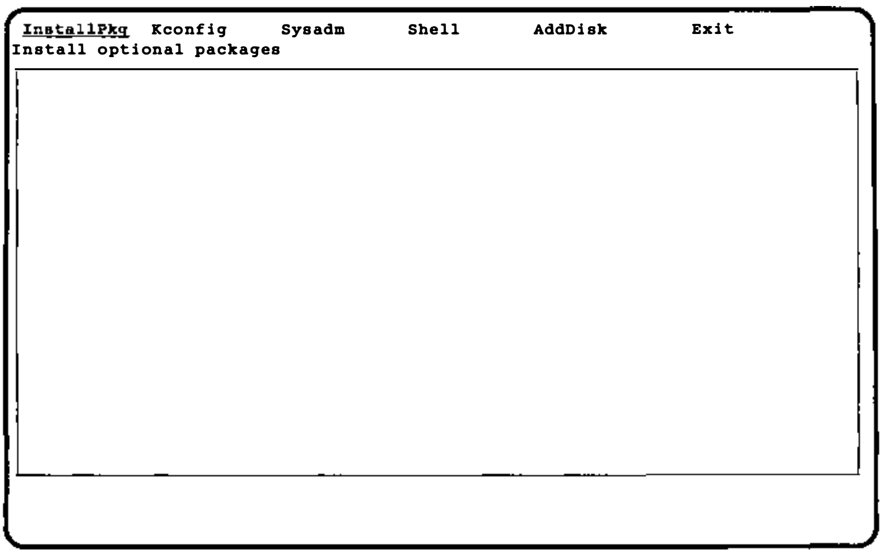
1. Type in the name you want to give your system. Be sure to give your computer a unique name if you plan to use it in a network.
2. When you have typed in the name you want, press ESC. The system displays:

   ```
   Press Y to confirm, N to cancel, E to continue editing
   ```

3. Type y when you are ready to exit the form.

4.7 Configuring the Software for Use

The system then gives you the opportunity to perform some optional software configuration. The system displays information screens describing the configuration process and the software package you are installing; then the configuration menu is displayed. Your screen will look similar to this:



```
InstallPkg Kconfig     Sysadm      Shell       AddDisk       Exit
Install optional packages
```

The configuration menu provides the following options:

- Select `InstallPkg` to install optional INTERACTIVE UNIX System subsets and extensions or other software packages. `installpkg` is a subcommand of the `sysadm` system administration program. Refer to section 6.1, "Installing Optional INTERACTIVE Subsets and Extensions" for a list of the subsets and extensions and for details about using `installpkg`. Refer to section 5 of "System Administration for New Users of the INTERACTIVE UNIX Operating System" for a general description of the `sysadm` program.
- Select `Kconfig` to run the `kconfig` program, which is used to configure, build, and install a new kernel for the operating system. You will need to do this when you add subsets and extensions that contain drivers for different physical devices, such as printers and networking cards, and to "tune" certain kernel parameters so that they are optimal for your particular system.

  ☛ Note that you must use the `Installpkg` option to install the Kernel Configuration subset before you attempt to use the `Kconfig` option.

Refer to section 7 of the "INTERACTIVE UNIX Operating System Maintenance Procedures" for a discussion of the `kconfig` program.

- The `sysadm` program can be used to perform most system administration tasks, such as installing additional software packages, adding new users, mounting and unmounting file systems, backing up files, and many more. Select the `sysadm` option to run this program. Section 5 of "System Administration for New Users of the INTERACTIVE UNIX Operating System" contains a general description of the `sysadm` program.
- Select the `Shell` option to run a shell, the standard interface between the user and the INTERACTIVE UNIX System. Select this option if you need to perform system configuration tasks for which you do not want to use `sysadm`. After finishing with the shell, type `exit` or press CTRL d to return to the menu.
- Select `AddDisk` if you want to add a second (or additional) fixed disk to your system. The `AddDisk` option calls `addhd`, which is a subcommand of the `sysadm` system administration program. Refer to section 4.8 for an example of adding a second fixed disk to your system.
- Select `Exit` when you have configured the system to your satisfaction.

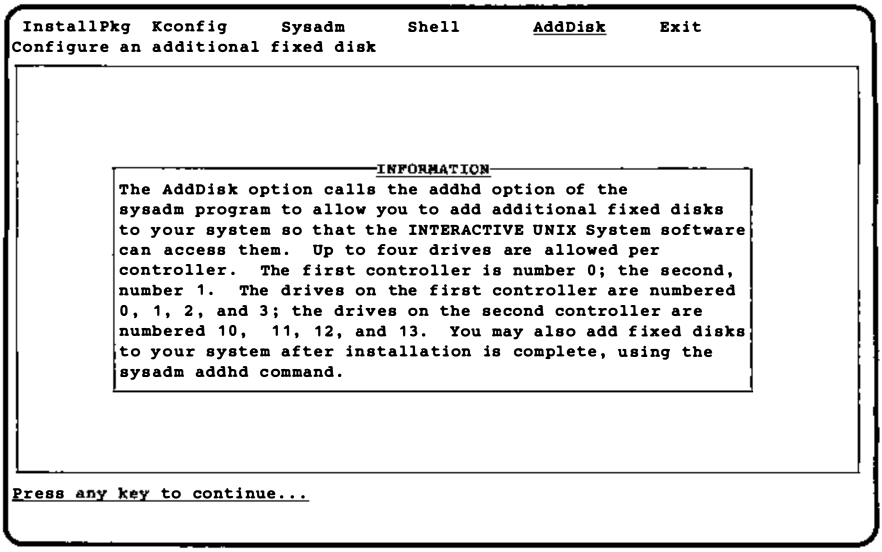
The system then shuts down automatically and waits for you to reboot it. Your system is now set up and ready to use.

4.8 Adding a Second Fixed Disk

You may add a second (or additional) fixed disk to your system configuration at the time of installation, or you may add disks at a later time using the `sysadm addhd` option.

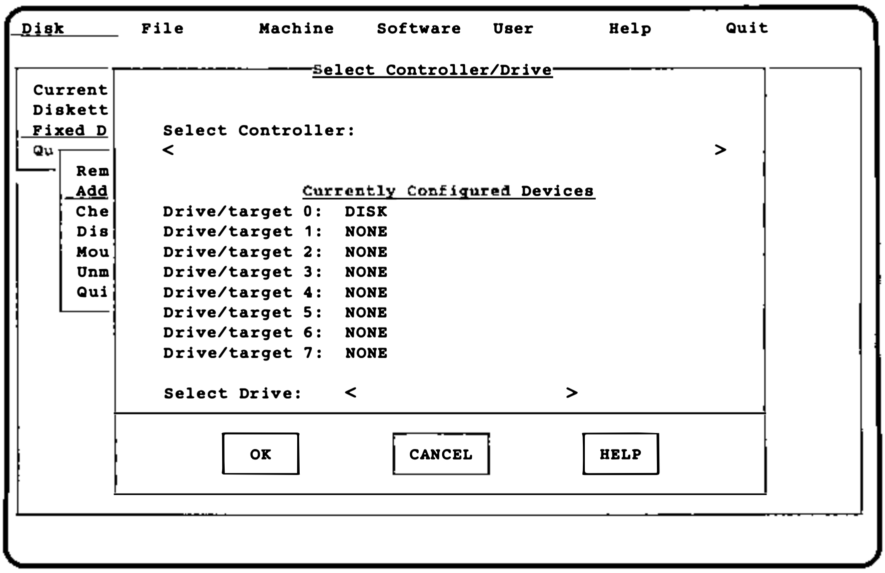
Note that from this point on, the `sysadm` program will have an enhanced look and feel. If you are unfamiliar with the features of the enhanced `sysadm` program, you can get on-line help by pressing the F1 key once or twice to access the Help Index. The Help Index is a list of topics for which help is available. TAB to a topic and press ENTER to view a help screen.

1. To add a second (or additional) fixed disk to your system configuration at the time of installation, select the `AddDisk` option from the configuration menu. Your screen will look similar to this:



```
Configure an additional fixed disk

                        INFORMATION
The AddDisk option calls the addhd option of the
sysadm program to allow you to add additional fixed disks
to your system so that the INTERACTIVE UNIX System software
can access them.  Up to four drives are allowed per
controller.  The first controller is number 0; the second,
number 1.  The drives on the first controller are numbered
0, 1, 2, and 3; the drives on the second controller are
numbered 10,  11, 12, and 13.  You may also add fixed disks
to your system after installation is complete, using the
sysadm addhd command.

Press any key to continue...
```

2. Press any key. The system displays a screen reminding you that you must format a fixed disk before you add it to the system. If you have not yet formatted your disk, exit the installation and format it before proceeding. If you are ready to proceed with installation, select the OK button at the bottom of the form. Your screen will look similar to this:

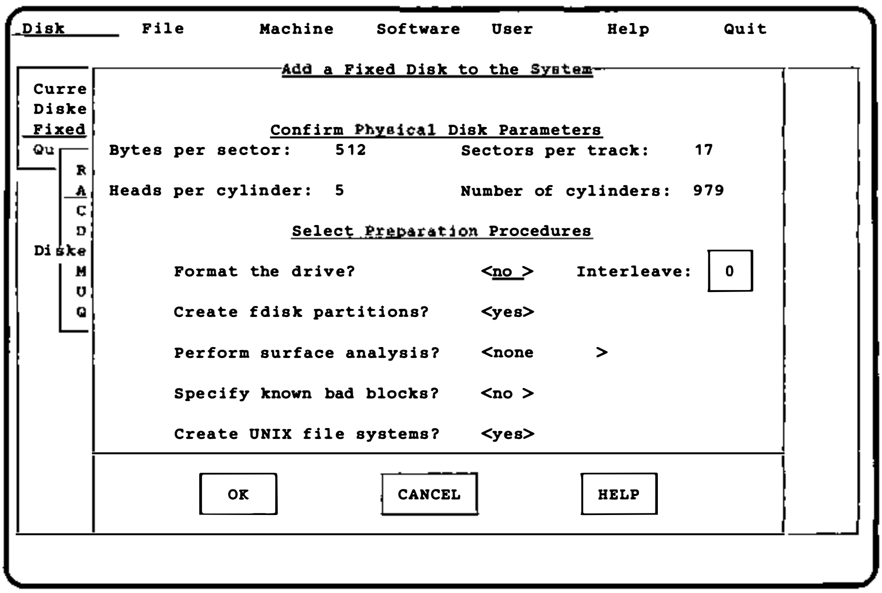


```
 Disk        File       Machine     Software    User        Help        Quit

┌────────┬─────────────────Select Controller/Drive─────────────────┬──────────┐
│Current │                                                         │          │
│Diskett │                                                         │          │
│Fixed D │   Select Controller:                                    │          │
│Qu ┌────┤   <                                             >       │          │
└───┤ Rem│                                                         │          │
    │ Add│               Currently Configured Devices              │          │
    │ Che│   Drive/target 0:   DISK                                │          │
    │ Dis│   Drive/target 1:   NONE                                │          │
    │ Mou│   Drive/target 2:   NONE                                │          │
    │ Unm│   Drive/target 3:   NONE                                │          │
    │ Qui│   Drive/target 4:   NONE                                │          │
    └────┤   Drive/target 5:   NONE                                │          │
         │   Drive/target 6:   NONE                                │          │
         │   Drive/target 7:   NONE                                │          │
         │                                                         │          │
         │   Select Drive:     <                   >               │          │
         ├─────────────────────────────────────────────────────────┤          │
         │      [ OK ]          [ CANCEL ]          [ HELP ]       │          │
         └─────────────────────────────────────────────────────────┘          │
└─────────────────────────────────────────────────────────────────────────────┘
```

3. In the first field, press the spacebar to view a list of the controllers currently configured on this system. Use the up and down arrow keys to move to the one you want, then press $\boxed{\text{ENTER}}$ to select it.

4. In the next field, press the spacebar to view a list of the fixed disk drives attached to the controller you selected. Move to the drive you want, then press $\boxed{\text{ENTER}}$ to select it.

5. Select the OK button at the bottom of the form. Your screen will look similar to this:



```
                     Add a Fixed Disk to the System

                   Confirm Physical Disk Parameters
Bytes per sector:    512          Sectors per track:     17

Heads per cylinder:  5            Number of cylinders:   979

                   Select Preparation Procedures

     Format the drive?              <no >   Interleave:  0

     Create fdisk partitions?       <yes>

     Perform surface analysis?      <none      >

     Specify known bad blocks?      <no >

     Create UNIX file systems?      <yes>
```

6. The first section of the form, `Confirm Physical Disk Parameters`, may or may not be editable, depending on the controller on your system. If your controller allows these parameters to be changed, the field will be editable. However, you should only edit this section of the form if the values listed are incorrect. Refer to the manufacturer's specifications included with your drive to verify the values shown. Refer to Appendix A for more information about fixed disk parameters and the INTERACTIVE UNIX System.

7. The next section of the form, `Select Preparation Procedures`, contains the following fields:

   `Format the drive?`

   Toggle the spacebar to select `yes` or `no`. Select `yes` to format the disk. Remember that formatting destroys all data on the entire disk. If you want to reformat the disk but have data you want to preserve, select `CANCEL` to stop this procedure, back up the data, then restart this procedure.

   `Interleave:`

   This field cannot be edited directly. If you decide to format your disk, another menu will be displayed, which allows you to change the interleave factor.

`Create fdisk partitions?`
Use the spacebar to select `yes` or `no`. Select `yes` if you want to create more than one partition on your disk. For example, if you plan on having more than one operating system on your disk, you will need to create a partition for each operating system. In this case, select `yes` and follow the on-line instructions. If the INTERACTIVE UNIX System Operating System is the only operating system you plan to install, you only need one partition. Press $\boxed{\text{ENTER}}$ to make your selection.

`Perform surface analysis?`
Toggle the spacebar to `read-write`, `read-only`, or `none`. You should normally perform some kind of surface analysis.

`Specify known bad blocks?`
If the manufacturer of your disk supplied a defect list for your drive, select `yes` to enter them into the defect table on the disk. This allows the system to avoid writing data to these bad blocks. You may also select `yes` if you just want to inspect the defect table. You may select `CANCEL` to leave the form at any time.

`Create UNIX file systems?`
If you want to divide the INTERACTIVE UNIX System File System partitions into file systems, such as `root` and `tmp`, select `yes` in this field.

8. When you have completed the form, select the `OK` button at the bottom of the form. The system will now perform all requested operations. When all operations are successfully completed, the fixed disk is added to the system.

4.9 Getting the System Ready to Use

After completing the setup of your system, you should establish accounts for your users. If you are familiar with the INTERACTIVE UNIX Operating System, all system administration functions can be run from the installation menu using the `sysadm` option. If you are new to the INTERACTIVE UNIX Operating

System, refer to section 5 of "System Administration for New Users of the INTERACTIVE UNIX Operating System" for a general overview of the `sysadm` program and refer to section 6.4.1, "Adding a New User," in that same document to find out how to create login accounts.

You may also need to customize your kernel. By default, the INTERACTIVE UNIX Operating System is optimized to support a system that has only 4 MB of RAM installed. Your system may have more memory available. (The total memory installed is reported during the boot procedure.) It is not necessary to reconfigure the kernel during the initial setup, but if your system has more than 4 MB of RAM installed, you will want to do it before users begin to work on your system to make efficient use of all the memory available on your system. Section 7.3.5.2 of "INTERACTIVE UNIX Operating System Maintenance Procedures" explains how to use the `kconfig` utility to change the default parameters for memory size. (Note that you cannot use the `kconfig` utility until you have installed the Kernel Configuration optional subset.)

After installation is complete, you should customize your disk driver configuration for your disk configuration. It is a good idea to use `kconfig` to configure the High Performance Device Driver (HPDD) because it will significantly speed up the boot process and for some controllers will allow you to run the controller in its native mode, taking advantage of all its features. Refer to section 8.2 of "INTERACTIVE UNIX Operating System Maintenance Procedures" for more information about the HPDD.

Customization may not be required if you plan to use only one controller, but you *must* customize your disk driver configuration if you plan to use multiple controllers, a SCSI tape drive, or a RAM disk. Refer to sections 7 and 8 of "INTERACTIVE UNIX Operating System Maintenance Procedures" for more information.

You have now completed the initial setup for your machine. After you have established a login account, you may log in to the system. The system is now fully operational.

If you are new to the INTERACTIVE UNIX System and are not familiar with any other UNIX-based system, you should already have read the "INTERACTIVE UNIX Operating System Primer." Refer to "System Administration for New Users of the INTERACTIVE UNIX Operating System" to learn how to install user accounts, back up and maintain files on the system, install and configure printers and other hardware devices, and tailor the system to match your requirements.

If you are experienced with the INTERACTIVE UNIX System or other UNIX-based systems, refer to the "INTERACTIVE UNIX Operating System Maintenance Procedures" for more technical information.

5. SHUTTING DOWN AND REBOOTING THE SYSTEM

5.1 Shutting Down the System

The INTERACTIVE UNIX Operating System is a *multi-tasking* system. A multi-tasking system can run many different processes (programs) at the same time. For example, you may be editing a file at the same time that another file is being printed on your printer. When you are ready to turn off your computer, you must arrange to have the system complete all of the tasks that are currently running. This is accomplished with a system maintenance procedure called `shutdown`. The `shutdown` program gracefully terminates the tasks that are currently executing before halting the system. You can safely turn off the computer when `shutdown` has finished. If you do not run the `shutdown` program, you may lose data and cause damage to your file system.

The `shutdown` program can be initiated in one of two ways:

1. Use the `powerdown` administrative login.
2. Execute the `shutdown` command.

5.2 Using the `powerdown` Administrative Login

When you are ready to turn your machine off, you may bring the system down with the `powerdown` administrative login.

1. Log out of your ordinary user account.
2. Log in to the system with the `powerdown` user ID.

   ☛ Note that you must know the `powerdown` password if one has been set.

3. Once you have successfully logged in to the system using the `powerdown` login, the system automatically executes the `shutdown` program. The system displays a screen similar to this:

```
login:  powerdown
Password:
UNIX System V/386 Release 3.2
Copyright (c) 1984, 1986, 1987, 1988 AT&T
Copyright (c) 1987, 1988 Microsoft Corporation
All rights reserved
Once started, a powerdown CANNOT BE STOPPED.
Do you want to start an express powerdown [y, n, ?, q]
```

4. If you are ready to bring the system down, type `y`. The system responds:

```
Shutdown started  Wed Jun  3 17:31:44 PDT 1987

Broadcast message from root (console) on plato
Wed Jun  3 17:31:44 PDT 1987
THIS SYSTEM IS BEING SHUT DOWN NOW!!!
Log off now or risk your files being damaged.

INIT: New run level 0
The system is coming down, please wait.
System services are now being stopped.
Stopping process accounting.

The system is down
Press any key to re-boot.
```

5. When the `Press any key to re-boot` message appears, the computer may be turned off.

5.3 Using the `shutdown` Command

To execute the `shutdown` command manually, you must log in to the system using the `root` login. When you are logged in as `root`, you will see the # prompt:

```
login:  root
Password:
#
```

If you are logged in to the system as `root` and want to execute the `shutdown` command manually, follow these instructions:

1. Make the `root` directory your current directory by executing this command at the # prompt:

   ```
   # cd /
   ```

2. Run the `shutdown` program with this command:

   ```
   # shutdown
   ```

   The system will automatically generate a message on every terminal currently in use to warn users that the system is being shut down. The message will look something like this:

   ```
   THIS SYSTEM IS BEING SHUTDOWN !!!
   Log off now or risk your files being damaged.
   ```

   The system will wait 1 minute to give users a chance to exit editors and save files before the system goes down.

3. Your screen will then look similar to this:

   ```
   Do you want to continue (y or n)
   ```

   Type `y`. Shutdown will then proceed. (If you do not want to be prompted at this point while shutting down the system, you can use the `-y` option when typing the `shutdown` command.)

4. If you wish to give users a different warning period before the system comes down, run the `shutdown` program using the `-g` option:

   ```
   # shutdown -gtime
   ```

   In actual use, *time* is replaced by the number of seconds you wish to have elapse before the system is halted. It is a good idea to allow at least 2 minutes (120 seconds) to elapse before the system is brought down. For example:

   ```
   # shutdown -g120
   ```

   The system automatically runs `shutdown`. A screen similar to the one generated by the `powerdown` procedure will display.

5. When the `Press any key to re-boot` message appears, the computer may be turned off.

5.4 Rebooting the System

You are now ready to begin using the INTERACTIVE UNIX Operating System. This section explains how to reboot the system if you have turned the computer off or run `shutdown`.

If you would like to continue exploring the INTERACTIVE UNIX Operating System and have already shut down your system, you must reboot the system and log in with your user ID.

To reboot the system, use this procedure:

1. Be sure there is no diskette in the diskette drive. If you have turned off the computer, turn on the power. The INTERACTIVE UNIX Operating System is automatically booted from the fixed disk.

2. If your computer is still turned on, either **1**) turn it off and then turn on the power again or **2**) press any key.

   The message `Booting the INTERACTIVE UNIX Operating System` will display, and the system will automatically reboot.

6. INSTALLING OPTIONAL SOFTWARE

6.1 Installing Optional INTERACTIVE Subsets and Extensions

After you have installed the Core subset, you will probably want to install one or more of the optional subsets and extensions delivered with your system. Depending on your requirements and the size of your fixed disk, select and install only those subsets that are necessary for your daily use. You can do this directly from the final installation menu using the `InstallPkg` option.

Subsets with special installation requirements have their own subsection here; read the associated section before attempting to install those subsets. Some subsets depend upon the presence of another subset to function properly. When installing or updating one of the subsets with a known dependency, be sure to install the necessary subset first. The following table lists the dependencies.

Optional Subsets	
Subset	*Dependencies*
Basic Networking	none
File Management	none
Help Utilities	none
Kernel Configuration	none
Spell Utilities	File Management
Terminal Utilities	none
STREAMS Facilities	Kernel Configuration
XENIX File System	Kernel Configuration
2 Kilobyte File System Utility Package	Kernel Configuration
Very Fast File System	Kernel Configuration
Additional Drivers	Kernel Configuration
User's Manual Entries	none

Optional Extensions	
Extension	*Dependencies*
Programmer's Manual Entries	none
International Supplement	File Management Kernel Configuration Basic Networking
VP/ix Environment Core	File Management
VP/ix Environment Configuration	Kernel Configuration VP/ix Environment Core
VP/ix Environment MS-DOS	File Management VP/ix Environment Core
TEN/PLUS* Environment	none
INTERACTIVE Software Development System	Kernel Configuration File Management
INTERACTIVE TCP/IP	Kernel Configuration STREAMS Facilities
INTERACTIVE Network Drivers	Kernel Configuration STREAMS Facilities
INTERACTIVE NFS Extension	Kernel Configuration INTERACTIVE TCP/IP STREAMS Facilities
INTERACTIVE NIS (formerly distributed as INTERACTIVE Yellow Pages*)	Kernel Configuration INTERACTIVE TCP/IP INTERACTIVE NFS Extension STREAMS Facilities
INTERACTIVE Text Processing Workbench	Terminal Utilities File Management
INTERACTIVE X11 Runtime System	Kernel Configuration STREAMS Facilities
INTERACTIVE X11 Development System	Kernel Configuration STREAMS Facilities INTERACTIVE Software Development System INTERACTIVE X11 Runtime System

Optional Extensions, Continued	
Extension	*Dependencies*
INTERACTIVE Motif* Development System	Kernel Configuration STREAMS Facilities INTERACTIVE X11 Runtime System INTERACTIVE X11 Development System
INTERACTIVE Looking Glass* Desktop Manager	Kernel Configuration STREAMS Facilities INTERACTIVE X11 Runtime System
INTERACTIVE Ported Netware	INTERACTIVE Network Drivers
INTERACTIVE SMB/ix* Extension	Kernel Configuration STREAMS Facilities INTERACTIVE TCP/IP
INTERACTIVE Network Connection Facilities	Kernel Configuration STREAMS Facilities
INTERACTIVE Security* Extension	Kernel Configuration **

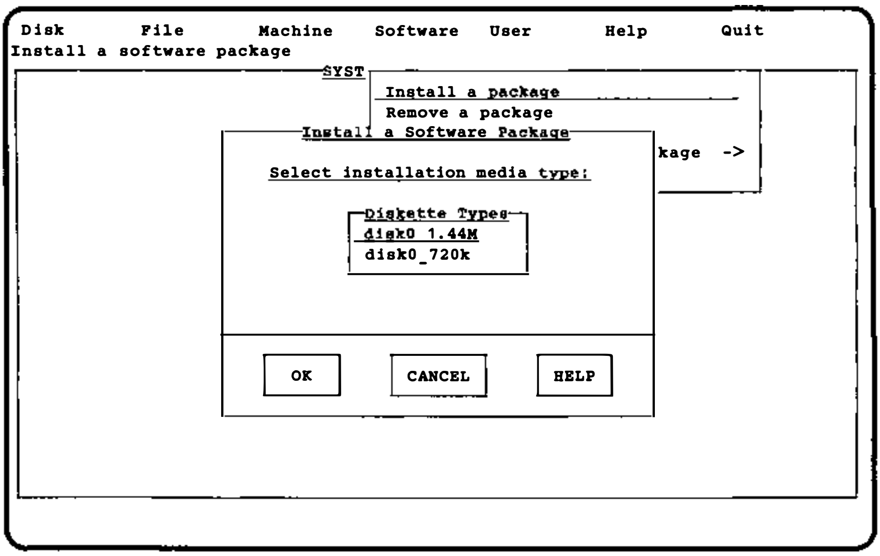
**Note that the installation of the INTERACTIVE Security Extension will fail if Basic Networking, the INTERACTIVE Network Connection Facilities, or the INTERACTIVE NFS Extension is installed.

Installation of a particular subset will fail if the subset it depends on is not already installed.

Use the final installation menu to install any optional subsets and extensions you want. (If you decide to add packages at a later date, you can use the `sysadm` program. You may access the menu by logging in to the system with the `sysadm` user ID, or you may use the `sysadm` command if you are already logged in to the system.)

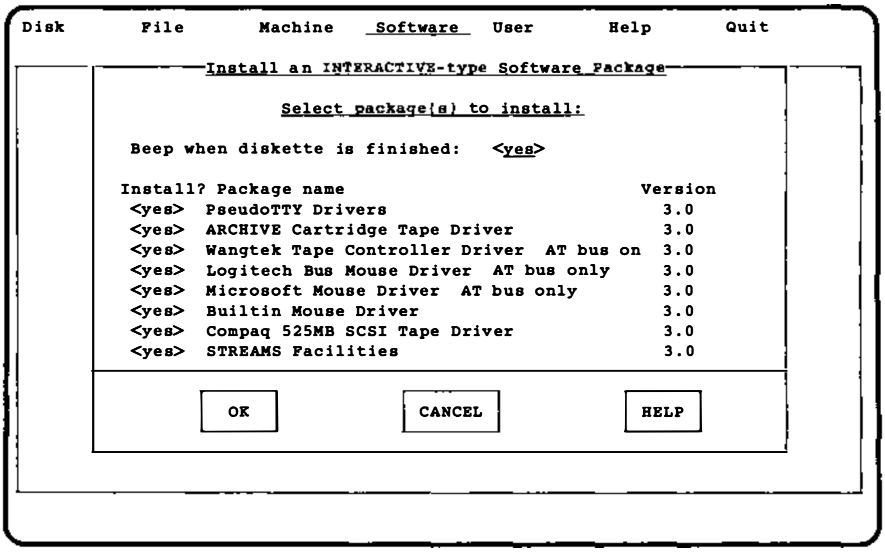
☛ If you are installing software that contains its own `sysadm` scripts for configuration, you must exit and reenter the `sysadm` program before these scripts will appear as options on the appropriate `sysadm` menu.

Use the `Installpkg` option or the `sysadm Install a Package` option under `Software`. Your screen will look similar to this:



```
Disk        File        Machine     Software    User        Help        Quit
Install a software package
                                   SYST
                                        Install a package
                                        Remove a package
                         Install a Software Package
                                                                 kage  ->
                      Select installation media type:

                               Diskette Types
                               disk0 1.44M
                               disk0_720k


                      OK            CANCEL            HELP
```

1. Select the type of diskette you are using for the installation, then select the OK button at the bottom of the form. You will be prompted to insert the diskette, then your screen will look similar to this:



```
Install an INTERACTIVE-type Software Package

Select package(s) to install:

Beep when diskette is finished:   <yes>

Install? Package name                                  Version
 <yes>   PseudoTTY Drivers                              3.0
 <yes>   ARCHIVE Cartridge Tape Driver                  3.0
 <yes>   Wangtek Tape Controller Driver  AT bus on      3.0
 <yes>   Logitech Bus Mouse Driver  AT bus only         3.0
 <yes>   Microsoft Mouse Driver  AT bus only            3.0
 <yes>   Builtin Mouse Driver                           3.0
 <yes>   Compaq 525MB SCSI Tape Driver                  3.0
 <yes>   STREAMS Facilities                             3.0
```

2. In the first field, use the spacebar to specify whether you want the system to beep when the diskette is finished.

3. The next part of the form lists the packages that can be installed on your system. By default, all packages listed have a `yes` in the corresponding field. To select a particular package for installation, press the spacebar to change the field to `no` for each package you do *not* want to install. Note that this form can be scrolled if there are more packages available than can be listed on the screen at one time.

4. Select the `OK` button when you have finished editing the form.

5. The system now displays a list of the files being installed for this software package. If the software package requires that a new kernel be built, your screen will display a message directing you to rebuild a kernel using `kconfig`. If you intend to install additional packages, you should install them before rebuilding the kernel.

6.2 Installing XENIX Software

The INTERACTIVE UNIX Operating System provides full support for applications written to run on the XENIX System V operating system. (This includes XENIX 286 and 386 `x.out` executable files.) An application written for XENIX may be installed and run under the INTERACTIVE UNIX Operating System with no loss in functionality.

To install a XENIX application package, follow the directions that accompany the application. Most applications require that the person installing the package have `root` privileges.

To remove a XENIX package from an INTERACTIVE UNIX System file system, use the `rm` command. (Refer to *rm*(1).)

☛ Most XENIX applications use the XENIX `custom` command during installation and removal of the package. For your convenience, this command is included with the INTERACTIVE UNIX Operating System. You can use the `Shell` option on the final installation menu to temporarily leave the installation program and run the `custom` command. Based on the application, you may be able to use the `custom` command to install application updates, customize the application, or remove it from the system.

6.3 Installing Other Software From Commercial Vendors

Many application packages are currently available for the INTERACTIVE UNIX Operating System. (Contact INTERACTIVE Systems Corporation or your INTERACTIVE products distributor for a catalog.) If a particular software package cannot be installed as described, follow the installation instructions provided with the application software or contact your software vendor.

Appendix A: FIXED DISKS AND CONTROLLERS

1. PHYSICAL COMPONENTS OF A FIXED DISK

If you are an experienced computer user, you may want to skip this discussion of the basic components of fixed disks.

One of the more important pieces of hardware in your system is the *fixed disk* drive on which the INTERACTIVE UNIX Operating System is installed. A fixed disk drive is a permanent, nonremovable storage device for data that is connected to your computer. Fixed disks are produced by many different manufacturers and can range in size from 10 or 20 MBs to multiple Gigabytes.

The documentation supplied by the manufacturer of your fixed disk should describe the physical structure of your disk. A fixed disk may be described by the number of *heads* (typically a number between 4 and 16), *cylinders* (usually a number between 100 and 1630), and *sectors* per *track* (usually a number between 17 and 63) that are available on it. These parameters vary from disk manufacturer to disk manufacturer. Each different set of parameters is associated with a number called a *disk type*. You are generally asked to supply a disk type when you run your hardware manufacturer's *setup* program.

A disk drive has a number of *disks* or *platters*, which are stacked together somewhat like a stack of phonograph records. There are usually between 4 and 16 of these surfaces. Each of the recording surfaces in the drive has its own recording *head* (similar to a needle on the record). Each head reads and writes data that is stored on the circular *tracks* on each disk surface. Each track is divided into 17 or more *sectors*, which are accessed as units. Data is read and written in sectors. The recording heads are bound together and move in unison. All the tracks that fall under the recording heads at any one point in time are called a *cylinder*. To save access time, data are generally written up or down all the tracks on the cylinder before the heads are moved to a new cylinder. Figure 1 illustrates the structure of a typical fixed disk.

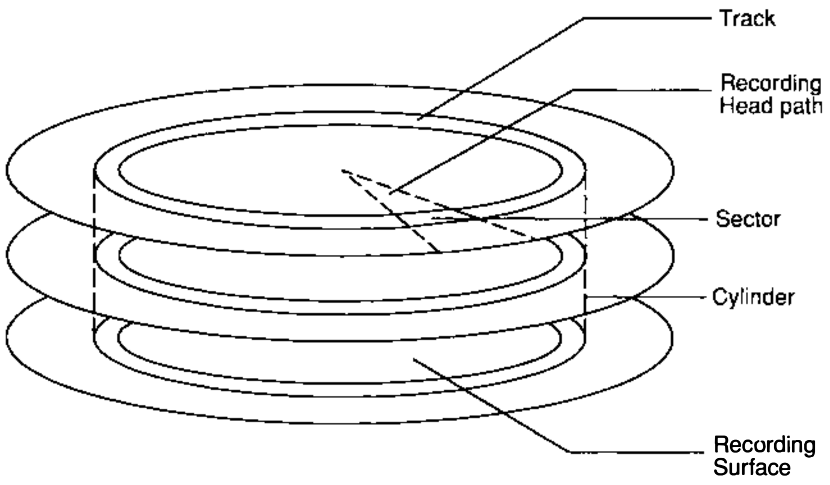


Figure 1. Structure of a Typical Fixed Disk

2. FIXED DISK PARAMETERS AND THE INTERACTIVE UNIX SYSTEM

During installation the INTERACTIVE UNIX Operating System attempts to determine the parameters of your fixed disk. These parameters should be correct for your system *even though* they *may not* match the true physical geometry (parameters) of your disk. They may instead reflect a virtual geometry imposed on the disk in support of MS-DOS or a modification of the true geometry that takes disk-specific methods of handling known bad sectors into account. Change the value(s) found by the system during installation *only* if you are very sure that one or more of the parameters are incorrect and that you know the correct parameters for your disk. Certain disk controllers restrict the parameters that may be changed; if this is the case with your controller, the system will not allow you to change them.

You do *not* need to know the disk parameters to install the INTERACTIVE UNIX Operating System under the following conditions:

- You are able to choose a disk type in your manufacturer's *setup* program that exactly matches your disk's parameters.
- You are using a SCSI adapter.
- You are using an Adaptec* or a Western Digital ESDI controller or a controller with a similar interface.

3. FIXED DISK CONTROLLER INTERFACE TYPES

Another very important piece of hardware in your system is the fixed disk *controller*. All fixed disk drives require an associated piece of hardware, called a controller, in order to operate. This board is responsible for interpreting the instructions that the operating system sends to the fixed disk drives and ensuring that the fixed disks carry out those instructions.

There are two major types of controllers used in 386-based systems, SCSI controllers (often pronounced "scuzzy") and AT-based controllers. SCSI controllers are frequently referred to as "host adapters." Most SCSI controllers can be configured to support multiple fixed disk drives and tape drives. Most AT-compatible controllers can be configured to support one or two fixed disk drives (of the same interface type). Each AT-compatible controller and drive use a particular type of interface to communicate. AT-compatible controllers are available for drives using most popular types of interfaces, including ST-506, RLL, and ESDI.

The INTERACTIVE UNIX Operating System must be compatible with the controller or controllers installed on your system. A wide variety of fixed disk controllers and drives are currently supported on the INTERACTIVE UNIX System, and INTERACTIVE is continually adding support for new devices.

4. FIXED DISK CONTROLLER COMPATIBILITY

To determine if your controller is supported, refer to the release notes delivered with your INTERACTIVE UNIX Operating System. If your controller is a standard AT-type controller that is not listed as supported, you may still be able to use it. Carefully follow the instructions for installing the INTERACTIVE UNIX System and see if your system will boot. If it does not, you may have to use a different controller.

5. USING MULTIPLE FIXED DISK CONTROLLERS

This section assumes a knowledge of the functions and characteristics of fixed disks and controllers. Additional information is located in section 8, "HARDWARE COMPATIBILITY AND CONFIGURATION," of the "INTERACTIVE UNIX Operating System Maintenance Procedures."

Most 386-based computer systems use one fixed disk and one controller. To form a larger, more complex system, several different fixed disk drives and controllers may be configured together.

Several factors must be considered when more than one disk controller is to be configured into a system. A complete discussion of the interactions among all possible combinations of controllers is beyond the scope of this document, but some common problems are listed below. Consult the documentation that accompanied your controller or contact your controller hardware vendor to determine whether these problems may exist with your controllers.

- *Address spaces*
  The I/O addresses used for communicating with the controllers must not overlap. In addition, any memory shared between the controllers and the system must have non-overlapping addresses. Most controllers allow such addresses to be selected by means of jumpers or switches on the controller boards.

- *Interrupts*
  For AT-compatible controllers (ST-506, RLL, and ESDI), assign each to a unique interrupt level. Unfortunately, not all controllers allow you to select their interrupt levels. Consult your hardware manufacturer or the documentation that accompanied your controller to make sure that the additional controller you want to use can be configured to use a secondary interrupt. However, for MCA-bus controllers, the same interrupt level may be shared.

- *DMA channels*
  If DMA is used, each DMA controller must use a unique DMA channel. These can usually be selected with jumpers or switches.

- *BIOS interactions*
  If the controller boards include BIOS ROMs, the addresses of each must be unique. In addition, some BIOS ROMs will interfere with each other or with the system BIOS. A common problem is that which occurs when a SCSI BIOS conflicts with the

system's Fixed Disk BIOS, making it impossible to successfully boot from a disk attached to an AT-compatible controller if the SCSI board is installed.

INTERACTIVE UNIX System defaults for each type of controller are set up to correspond to those of the shipped boards, so no changes should be necessary in these if you are using controllers of different types. Be sure to consult your hardware vendor before attempting to use two controllers of the *same* type to ensure that they can be configured differently.

6. CUSTOMIZING YOUR SYSTEM TO MAXIMIZE THE DISK CONFIGURATION

Once you have installed the INTERACTIVE UNIX Operating System, you should customize the HPDD configuration to speed up the boot process and to maximize the performance throughput. Refer to section 8.2.4, "Configuring the High Performance Device Driver," in the "INTERACTIVE UNIX Operating System Maintenance Procedures" for details.

Appendix B: THE HARDWARE SETUP PROGRAM

1. INTRODUCTION

If your hardware vendor does not set up your machine for you, then after you have verified that your hardware configuration meets the minimum hardware requirements, you must run the system manufacturer's *setup* program. (The actual name of this program may be something other than "*setup*." Check the documentation that accompanied your hardware. Some *setup* programs are incorporated into the system's ROM BIOS and are invoked by pressing a specific combination of keys soon after the system is powered up.) Many different types of hardware can be combined to create a working INTERACTIVE UNIX System. The *setup* program informs your computer of the types of hardware components that are present and their characteristics. This information is stored permanently on the system.

1.1 Selecting a Fixed Disk Type

The *setup* program will ask you to select a disk type. This is usually a number between 1 and 42, although it depends on the manufacturer. Since there are many more disks available today than there are defined disk types, you may need to select the disk type that most closely matches your disk's number of heads, cylinders, and sectors per track. If there is no entry that lists the exact number of cylinders, choose an entry with the correct number of heads and sectors per track, and a smaller number of cylinders.

The INTERACTIVE UNIX Operating System attempts to make installation as simple as possible while still offering a great deal of flexibility in the types of hardware that you can use. The INTERACTIVE UNIX Operating System will either be able to determine the correct values from your controller, or it will use the values that you enter using your manufacturer's *setup* program as the defaults, and then allow you to supply the actual physical characteristics of the disk during installation. The *Boot* diskette will *automatically* determine the type of controller present in your system and use its first fixed disk as the boot device and location for the `root` file system.

If you want to boot from a disk attached to an AT-compatible controller, use the exact match of disk type (if available), or set the disk type equal to 1. In most cases (as long as the INTERACTIVE UNIX System is your only operating system), a disk type of 1 is sufficient for any drive. A disk type of 1 indicates the smallest number of heads, cylinders, and sectors per track possible, and all fixed disks have at least this configuration available. Adaptec's AT-compatible disk controllers that support more than 17 sectors per track use their own disk type information. When installing one of these controllers, be sure to use your *setup* program to set the disk type to `1`.

If you want to boot from a disk attached to a COMPAQ* controller, you *must* have an exact match for the disk type in your manufacturer's *setup* program.

If you want to boot from a disk attached to a SCSI adapter, use the manufacturer's *setup* program to configure your computer system so that *no* AT-compatible fixed disks are installed. Set the disk type to `0` (note that in some *setup* programs, this is equivalent to making the choice `Not installed`). The SCSI boot disk should be configured so that it is Logical Unit Number (LUN) 0 on SCSI Target ID 0. A SCSI *tape* device should be configured to be a higher Target ID number than any other disk drive present. Refer to the documentation that accompanied your SCSI disk drive or contact your hardware vendor to determine this information for your drive, whether or not it can be changed, and how to change it, if necessary.

For MCA machines, run the reference diskette or the equivalent program supplied by the manufacturer to properly set up the controllers and the machines.

Controllers other than those listed here may have other requirements; refer to your hardware manufacturer's installation instructions for such information.

INDEX

INTERACTIVE UNIX Operating System Maintenance Procedures

CONTENTS

INTERACTIVE UNIX* Operating System Maintenance Procedures

1. INTRODUCTION

The "INTERACTIVE UNIX Operating System Maintenance Procedures," together with "System Administration for New Users of the INTERACTIVE UNIX Operating System," in the *INTERACTIVE UNIX System Guide for New Users*, and the *INTERACTIVE UNIX System User's/System Administrator's Reference Manual*, provide you with detailed information on most aspects of installing, configuring, and maintaining an INTERACTIVE UNIX Operating System. Refer to the "Documentation Roadmap" in this guide for information about additional documentation.

1.1 Overview of This Document

This document is divided into 14 major sections:

1. INTRODUCTION
   This section provides a general overview of this document.

2. THE INTERACTIVE CHARACTER USER INTERFACE
   This section gives a general description of the INTERACTIVE Character User Interface System (CUI), which is the interface used by the `sysadm` system administration program and `kconfig`, the kernel configuration program. It explains how to get help at any time while using the system and how to use the menus and forms to perform tasks.

3. THE SYSTEM ADMINISTRATION PROGRAM
   This section gives a general description of how the system administration program is used to perform system administration tasks and describes how to access its menus using the `sysadm` command.

4. CREATING AND USING INTERACTIVE UNIX SYSTEM FILE SYSTEMS
   This section describes the procedures for creating and maintaining INTERACTIVE UNIX System file systems that are run from the command line.

5. FILE SYSTEM MAINTENANCE

   This section explains how to use `fsck` from the command line to check and repair file systems, how bad blocks are detected and handled, the fixed disk layout, how to recover from a system crash, and how to mount diskette-based file systems when using DOS-FSS. It also discusses the various file system types available in the INTERACTIVE UNIX Operating System.

6. BACKING UP AND RESTORING FILES

   This section describes how to back up individual files, directories, and file systems and how to restore them.

7. USING `kconfig` TO TAILOR YOUR SYSTEM KERNEL

   This section describes the `kconfig` program interface, and the options available through it for configuring your kernel. Using `kconfig`, you can add, modify, and remove drivers, add and remove facilities, add and modify default parameters for memory size, add and modify tunable system parameters, display the High Performance Device Driver configuration, configure the High Performance Device Driver, and build and install a new kernel. This section also explains what to do if the system does not boot.

8. HARDWARE COMPATIBILITY AND CONFIGURATION

   This section contains descriptions of the hardware requirements, software setup, configuration information, and possible conflicts for all the drivers included with the INTERACTIVE UNIX Operating System.

9. LP PRINT SERVICE ADMINISTRATION

   This section discusses the detailed workings of the LP print service system and describes how to customize the system for special printers or particular printing requirements.

10. ADDING MODEMS, PRINTERS, AND OTHER SERIAL DEVICES

    This section explains the basic networking procedures required to add serial devices to your system and the software configuration that is required.

11. BASIC NETWORKING ADMINISTRATION
    This section describes the administration of the Basic Networking utilities (primarily `uucp`) that allow your system to communicate with other UNIX System computers using either dial-up or hard-wired communication lines.

12. ELECTRONIC MAIL OPTIONS
    This section explains the electronic mail options available with the INTERACTIVE UNIX System and provides pointers to the information for setting up the mail.

13. TROUBLESHOOTING
    This section describes some of the most frequently asked questions that arise during the installation and configuration of the INTERACTIVE UNIX Operating System to help you to troubleshoot problems with your system.

14. Appendix: KERNEL ERROR MESSAGES
    This section lists the kernel panic, warning, and notice messages along with possible actions you can take.

2. THE INTERACTIVE CHARACTER USER INTERFACE

The `sysadm` and `kconfig` programs both make use of the INTERACTIVE Character User Interface (CUI). This system is designed to be as self-explanatory as possible. If you've never used it before, you should read, either in this guide or on-line, the short summary of how to get help at any time and how to use the menus and forms to perform tasks.

One way to learn about the system is to look at it on-line while reading about it here. You can look at the forms on-line without altering your system; no actions are taken until you explicitly "instruct" `sysadm` to do so. The system displays a warning screen and requires you to confirm your action before any potentially destructive changes are initiated. You can also run `sysadm` in a demonstration mode by typing:

```
sysdemo
```

In this mode, all `sysadm` actions are disabled, and you can safely explore the system.

2.1 Using the CUI Help Facility

You can press [F1] for on-line help at any time. If context-specific help is available, it appears. Pressing [F1] a second time accesses the general Help Index. If no context-specific help is available, the Help Index appears immediately. Use the up and down arrow keys or [PAGE-UP] and [PAGE-DOWN] to scroll through the Help Index. Press [ENTER] to select a topic.

There are also `HELP` buttons at the bottom of many forms. Use [TAB] or [BACK-TAB] to move to the `HELP` button, and press [ENTER] to obtain general help on the form you are viewing. Some help topics have more than one page. To move from page to page of each help topic, use [PAGE-UP] or [PAGE-DOWN]. You can exit from the help system at any time by pressing [ESC]. The following table summarizes how to use the help system:

Using the Help System	
ACTION	*KEY*
Display context-specific help	<F1>
Display the Help Index	<F1> when already in the Help System <F1> <F1> when you are *not* in the help system
Scroll through Help Index	up and down arrow keys
Page through Help Index	<PAGE-UP> or <PAGE-DOWN>
Select a Help Index topic	<ENTER>
Move to the `HELP` button	<TAB> or <BACK-TAB>
Activate the `HELP` button	<ENTER>
Page through help topics	<PAGE-UP> or <PAGE-DOWN>
Exit the help system	<ESC>

2.1.1 Hypertext

The help in this program is a hypertext system. The system provides one or more pages of help on a number of topics. Topics are connected to each other by means of links. Links are words that appear on-line in **bold** text. Three standard links appear at the bottom of each help window:

```
Exit    Go Back    Index
```

`Exit`	– always takes you out of the help system.
`Go Back`	– always takes you to the previous topic, if one exists.
`Index`	– always takes you to the Help Index.

Links in the text provide a path to additional information about a topic that you may or may not want to see. Press **TAB** or **BACK-TAB** to move from link to link. When the one you want is highlighted (active), press **ENTER** to follow it.

2.1.2 Links

When you follow a link, a new topic appears. If there is more than one page in the topic, use **PAGE-UP** or **PAGE-DOWN** to move through it. To return to the previous topic, press **TAB** or **BACK-TAB** to move to `Go Back` at the bottom of the help window and press **ENTER**. You can follow any number of links through this system. Using `Go Back` will retrace your steps exactly to the original point at which you entered the system.

2.2 Using INTERACTIVE CUI Menus and Forms

The INTERACTIVE CUI System is designed to make administration of the INTERACTIVE UNIX Operating System as easy as possible. Its major components are the bar menu at the top of your screen, pull-down menus, and forms.

2.2.1 The Bar Menu

Use the left and right arrow keys ← → to move from title to title of the bar menu. Press F1 for on-line help on the bar menu titles. Press ENTER to pull down a menu. To exit from the bar menu and return to the system prompt, move to `Quit` and press ENTER.

2.2.2 Pull-Down Menus

Use the up and down arrow keys ↑ ↓ to move from option to option, or type the first letter(s) of an option name. If there is an arrow (->) following the option, another menu appears when you press ENTER. If there is no arrow, a form appears when you press ENTER. Press F1 for on-line help on the menu items. To exit from a pull-down menu, you can either move to `Quit` (or type `q`) and press ENTER, or you can simply press ESC.

2.2.3 Forms

Forms are used to display or gather information needed to perform the task you selected from a pull-down menu. Forms contain several different kinds of fields. Some fields are used only to display information; you cannot press TAB or BACK-TAB to move to them or change them. Other fields allow you to edit them by making choices or typing information. Help is available for fields that you can edit. Press F1 for on-line help on these fields. To move from field to field and button to button, press TAB or BACK-TAB.

All forms have at least one button in a box at the bottom of the form. Most forms have these buttons:

```
OK      CANCEL      HELP
```

`OK`	confirms that you are satisfied with the information on the form and want your changes to take effect.

`CANCEL`	exits from the form without any changes taking effect.
`HELP`	provides help for the form.

To activate a button, press TAB or BACK-TAB to move to it and press ENTER.

2.2.4 Fields

Forms in this system contain a number of different kinds of fields. Some fields are lists that you can scroll through using the up and down arrow keys. Some lists are always visible, while others require you to press the spacebar to pop them up. Press ENTER while an item is highlighted to select it from the list. Help is available for fields that you can edit. Press F1 for on-line help on these fields.

< >	**Fields within angle brackets** < > allow you to use the spacebar to cycle through the choices for that field or to press the spacebar to pop up a list of values. Use the up and down arrow keys to scroll through the items in a list. Press ENTER to select one.
[ ]	**Fields within square brackets** [ ] are check boxes. Press the spacebar to "check" the box (an `X` appears). You can press the spacebar to toggle the `X` in or out of a field.
( )	**Fields within parentheses** ( ) are radio buttons. They usually occur in sets where a choice must be made between two or more mutually exclusive options. Press the spacebar to turn the radio button "on" and an asterisk (*) appears. If you turn on one radio button, the other choice or choices automatically go "off."

For a summary of the key strokes used in this system, see Using Bar Menus, Using Pull-Down Menus, Using Forms, and Special Keys on your Character User Interface Quick Reference Card.

3. THE SYSTEM ADMINISTRATION PROGRAM

3.1 Introduction

This section provides an overview of the system administration program. The system administration program is a standard feature of the INTERACTIVE UNIX Operating System and is used to perform commonly required system administration tasks. It is designed to be as easy to use and self-explanatory as possible. The `sysadm` program uses the INTERACTIVE Character User Interface (CUI) to display menus and forms. You may want to read the brief introduction to the CUI System in the previous section (or on-line) before attempting to use `sysadm`. Experienced UNIX System users can perform the system administration tasks from the command line rather than using the menu interface, if desired.

More detailed information about using these menus can be found in the appropriate sections of this document and in "System Administration for New Users of the INTERACTIVE UNIX Operating System."

☛ This document does not attempt to describe every possible maintenance procedure available through the system administration program. The system is easy to follow, and help is available on-line at any time by pressing $\boxed{\text{F1}}$.

`sysadm` can be used as a login account or as a utility program. A password should be assigned to `sysadm` when the system is installed.

You can also run `sysadm` in a demonstration mode by typing:

```
sysdemo
```

In this mode, all `sysadm` actions are disabled, and you can safely explore the system.

The following conventions apply when you are using the `sysadm` login, `sysadm` command, or any `sysadm` menu:

- You must be at the console terminal to log in as `sysadm`, and you must know its password.
- If you do not understand the options or what to put in to a field in a form displayed by the system, press $\boxed{\text{F1}}$ for *help* at any time. If context-specific help is available, it appears. Pressing $\boxed{\text{F1}}$ a second time accesses the general Help Index. If no context-specific help is available, the Help Index appears

immediately. You can exit from the help system at any time by pressing ESC.

- To exit from the bar menu and return to the system prompt, or to exit from a pull-down menu, use the arrow keys to move to `Quit` (or type `q`) and press ENTER. You can also exit from a pull-down menu by simply pressing ESC. To exit from a form without any changes taking effect, move to the `CANCEL` button and press ENTER. To exit from a form with changes taking effect, move to the `OK` button and press ENTER.

3.1.1 `sysadm` *Menu Bypass*

You may access a `sysadm` pull-down menu directly by typing `sysadm`, followed by the appropriate command. For example, typing:

```
# sysadm harddisk
```

displays the `sysadm Fixed Disk Management` menu.

Each `sysadm` pull-down menu contains a list of options and/or other menus. You may bypass the menus and access a known option directly by typing the option name as an argument to `sysadm`. For example, the `Diskette Management` menu contains an option called `Format a Diskette`. Access that option directly by typing:

```
# sysadm fmtdisk
```

If you use `sysadm` as a login, you may also supply the `sysadm` options described above, such as `fmtdisk`, on the command line. The following table lists all the menus and options and their abbreviations:

Menu or Form	Command
Disk	`disk`
Current Disk Usage	`diskuse`
Diskette Management	`diskettes`
Copy a Diskette	`cpdisk`
Erase a Diskette	`eradisk`
Format a Diskette	`fmtdisk`
Verify a Diskette	`verdisk`
File System Management	`filesys`
Check a Diskette File System	`chkfdfs`
Create a Diskette File System	`mkfdfs`
Mount a Diskette File System	`mntfdfs`
Unmount a Diskette File System	`umntfdfs`
Fixed Disk Management	`harddisk`
Remap Disk Defects	`badblks`
Add a Fixed Disk to the System	`addhd`
Check a Fixed Disk File System	`chkhdfs`
Display Fixed Disk Information	`lshd`
Mount a Fixed Disk File System	`mnthdfs`
Unmount a Fixed Disk File System	`umnthdfs`

Menu or Form	Command
File	`file`
Back Up or Restore Files	`backups`
Back Up Selected Files	`backup`
Archival/Incremental Backup	`fsback`
Restore Files	`restor`
File Information	`fileinfo`
Owner/Group	`filuse`
Age	`filage`
Size	`filsiz`

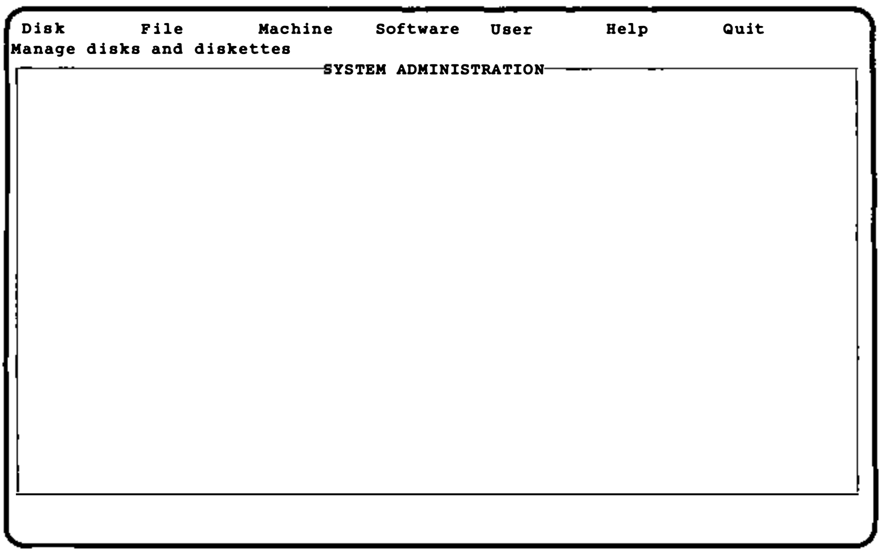
Menu or Form	**Command**
Machine	`machine`
Current Disk Usage	`diskuse`
Modem Initialization	`setmodm`
Printer Management	`lpmgmt`
Add a Printer	`addprt`
Delete a Printer	`delprt`
Modify a Printer	`chgprt`
Change Printer Status	`statprt`
List Printers	`lsprt`
Spooler Management	`statspl`
System Setup	`syssetup`
Display/Change System Configuration	`setconf`
Set System Passwords	`syspass`
Tty Management	`ttys`
Virtual Terminals	`chgvts`
Modify Tty Parameters	`chgtty`
Getty Line Management	`gettys`
Add	`addgetty`
Delete	`delgetty`
List	`lsgetty`
Modify	`chgetty`

Menu or Form	Command
Software	`software`
Install a Package	`addpkg`
Remove a Package	`delpkg`
List Installed Packages	`lspkg`
Mail System Setup	`setmail`
Basic Networking	`UUCP`
Test a UUCP Connection	`callsys`
Modem Initialization	`setmodm`
Dialcode Management	`dcode`
Add	`adddcode`
Delete	`deldcode`
List	`lsdcode`
Modify	`chgdcode`
Outgoing Connections	`device`
Add	`adddev`
Delete	`deldev`
List	`lsdev`
Modify	`chgdev`
Permissions Management	`perm`
Add	`addperm`
Delete	`delperm`
List	`lsperm`
Modify	`chgperm`
Poll Management	`poll`
Add	`addpoll`
Delete	`delpoll`
List	`lspoll`
Modify	`chgpoll`
Incoming Connections	`port`
Add	`addport`
Delete	`delport`
List	`lsport`
Modify	`chgport`
System Management	`system`
Add	`addsys`
Delete	`delsys`
List	`lssys`
Modify	`chgsys`

Menu or Form	Command
User (includes Group)	`user`
List Active Users	`whoson`
Group Management	`groups`
Add a New Group	`addgroup`
Remove a Group	`delgroup`
Change a Group Name	`chgroup`
List Groups	`lsgrps`
List All Groups	`lsgroup`
List System Groups	`lsSgroup`
List User Groups	`lsUgroup`
User Management	`users`
Add a New User	`adduser`
Remove a User	`deluser`
Modify User Information	`chguser`
Set 'Add User' Defaults	`usrdefs`
Set Available Shells	`shdefs`
Supplementary Groups	`supgrp`
List Users	`lsusrs`
List All Users	`lsuser`
List System Users	`lsSuser`
List Regular Users	`lsUuser`

3.2 The `sysadm` Menus

When you type the `sysadm` command, the bar menu is displayed. If you set a password for the `sysadm` login when you installed the system, you will be prompted for a password in the usual manner. The system displays a screen similar to this:



```
 Disk        File        Machine     Software    User        Help        Quit
Manage disks and diskettes
                              SYSTEM ADMINISTRATION
```

Each title on the bar menu represents a category of system administration tasks. Use the left and right arrow keys (or type the first letter(s) of a title) to move to a title. A short summary of that title is displayed under the bar menu. When you move to a title and press ENTER to select it, the system pulls down a menu that lists tasks that are specific to that option. When you select an option from a pull-down menu, the system either provides a form to view or fill in to perform the specified task, or the system displays another menu. If an arrow follows an option name, for example:

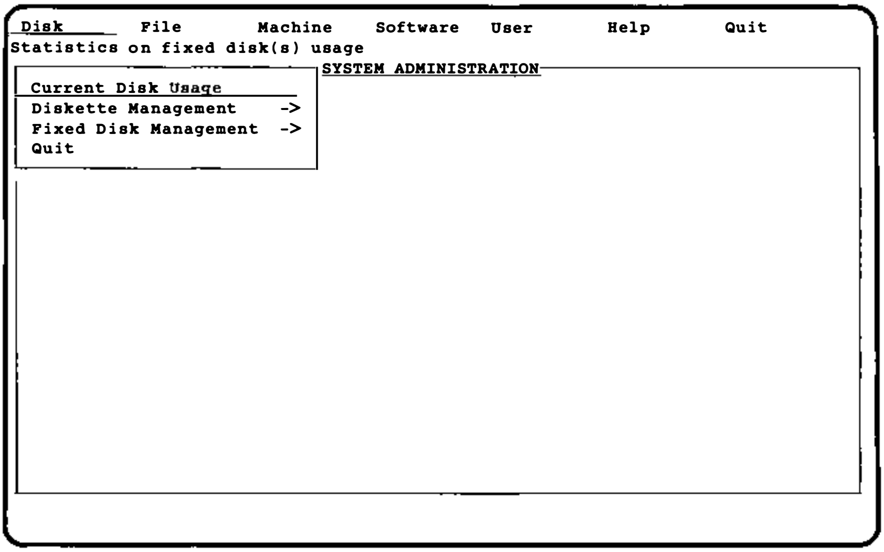
```
Diskette Management  ->
```

another menu appears when you press ENTER.

If you want to return to the previous menu at any point, type `q` or use the arrow keys to move to `Quit` on the menu, and then press ENTER.

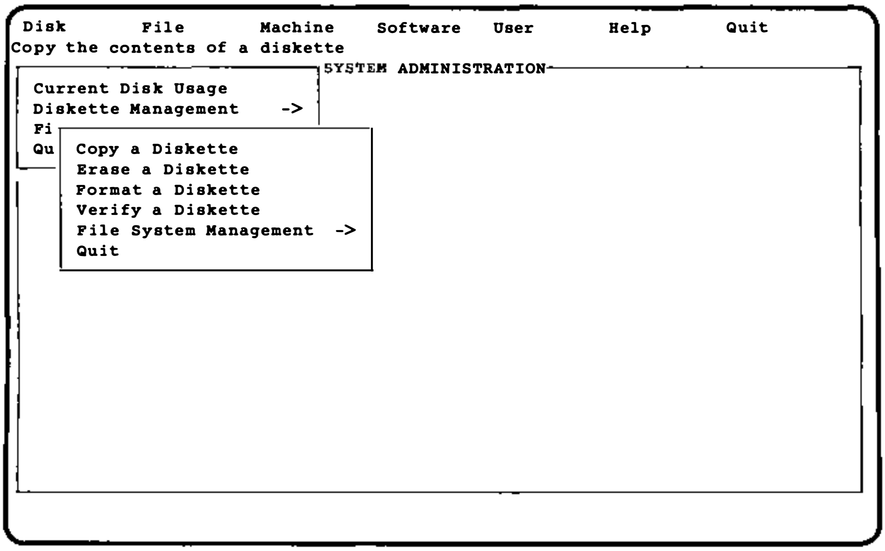
3.2.1 *The* `Disk` *Menu*

The `Disk` menu looks similar to this:



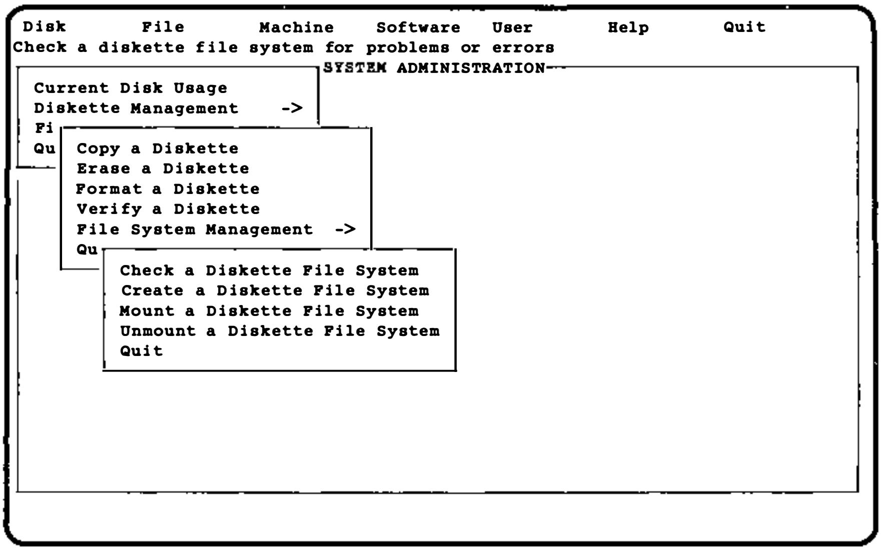
```
Disk        File       Machine     Software   User       Help       Quit
Statistics on fixed disk(s) usage
                                        SYSTEM ADMINISTRATION
 Current Disk Usage
 Diskette Management      ->
 Fixed Disk Management    ->
 Quit
```

The `Diskette Management` option displays a menu that looks similar to this:



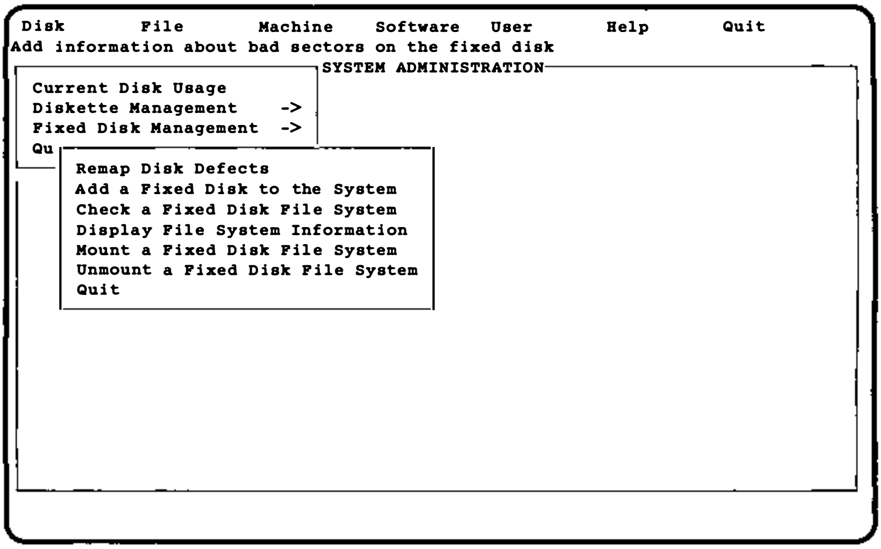
```
Disk        File       Machine     Software   User       Help       Quit
Copy the contents of a diskette
                                        SYSTEM ADMINISTRATION
 Current Disk Usage
 Diskette Management      ->
 Fi
 Qu   Copy a Diskette
      Erase a Diskette
      Format a Diskette
      Verify a Diskette
      File System Management  ->
      Quit
```

`File System Management` includes:



```
 Disk        File        Machine     Software    User        Help        Quit
Check a diskette file system for problems or errors
┌───────────────────────────┐SYSTEM ADMINISTRATION──────────────────────────┐
│ Current Disk Usage        │
│ Diskette Management    -> │
│ Fi┌───────────────────────────┐
│ Qu│ Copy a Diskette           │
└───│ Erase a Diskette          │
    │ Format a Diskette         │
    │ Verify a Diskette         │
    │ File System Management  ->│
    │ Qu┌──────────────────────────────┐
    └───│ Check a Diskette File System │
        │ Create a Diskette File System│
        │ Mount a Diskette File System │
        │ Unmount a Diskette File System
        │ Quit                         │
        └──────────────────────────────┘
```

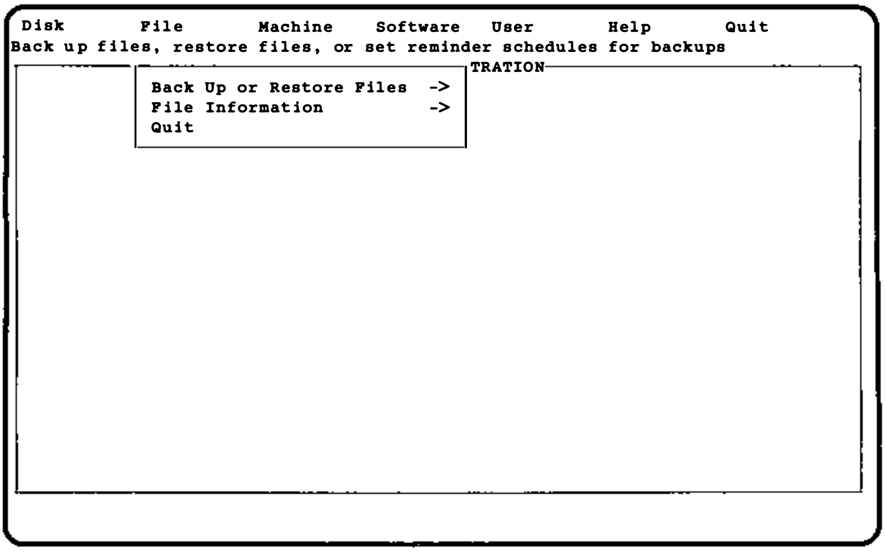
The `Fixed Disk Management` option displays a menu that looks similar to this:



```
 Disk        File        Machine     Software    User        Help        Quit
Add information about bad sectors on the fixed disk
┌───────────────────────────┐SYSTEM ADMINISTRATION──────────────────────────┐
│ Current Disk Usage        │
│ Diskette Management    -> │
│ Fixed Disk Management  -> │
│ Qu┌──────────────────────────────────┐
└───│ Remap Disk Defects               │
    │ Add a Fixed Disk to the System   │
    │ Check a Fixed Disk File System   │
    │ Display File System Information  │
    │ Mount a Fixed Disk File System   │
    │ Unmount a Fixed Disk File System │
    │ Quit                             │
    └──────────────────────────────────┘
```

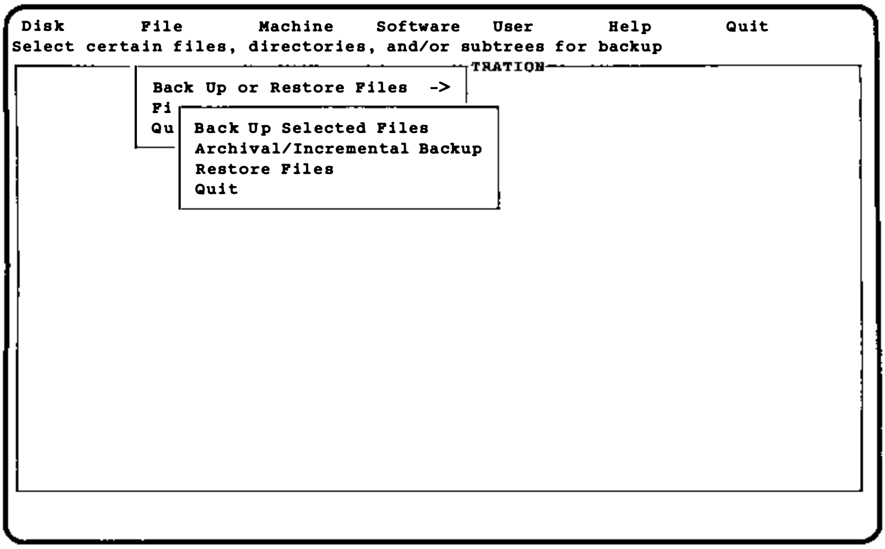
3.2.2 *The* `File` *Menu*

The `File` menu looks similar to this:



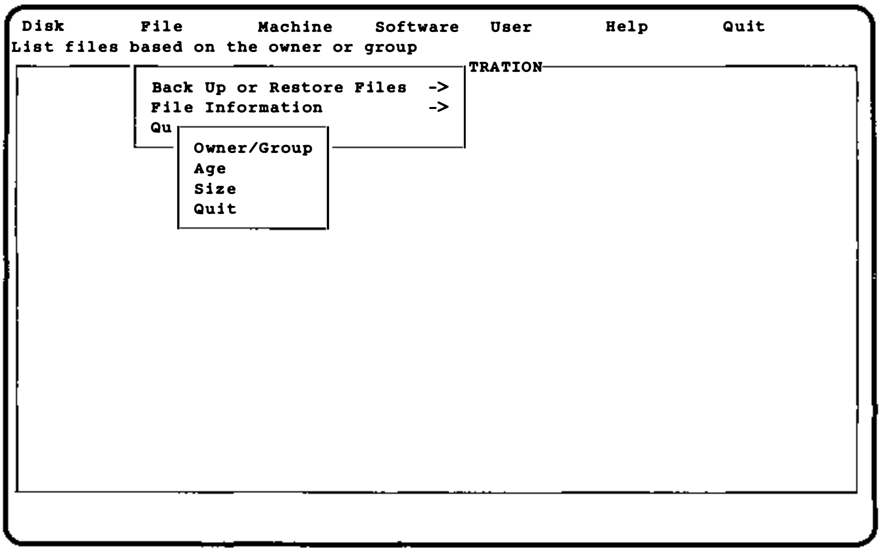
```
 Disk        File        Machine     Software    User        Help        Quit
Back up files, restore files, or set reminder schedules for backups
                                                  TRATION
             Back Up or Restore Files  ->
             File Information          ->
             Quit
```

The `Back Up or Restore Files` menu displays the following:



```
 Disk        File        Machine     Software    User        Help        Quit
Select certain files, directories, and/or subtrees for backup
                                                  TRATION
             Back Up or Restore Files  ->
             Fi
             Qu  Back Up Selected Files
                 Archival/Incremental Backup
                 Restore Files
                 Quit
```

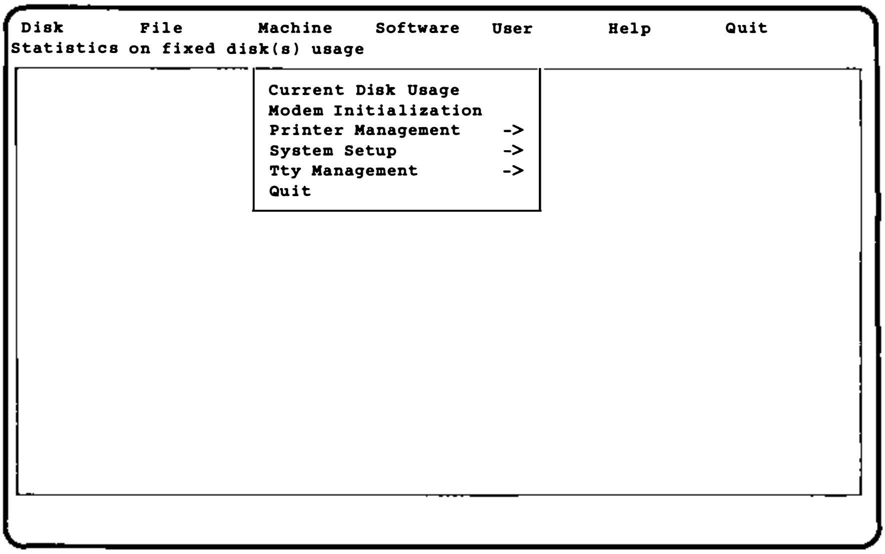
The `File Information` menu under `File` on the bar menu displays the following:



```
 Disk        File        Machine     Software   User        Help        Quit
List files based on the owner or group
                                                TRATION
             Back Up or Restore Files  ->
             File Information          ->
             Qu
                 Owner/Group
                 Age
                 Size
                 Quit
```

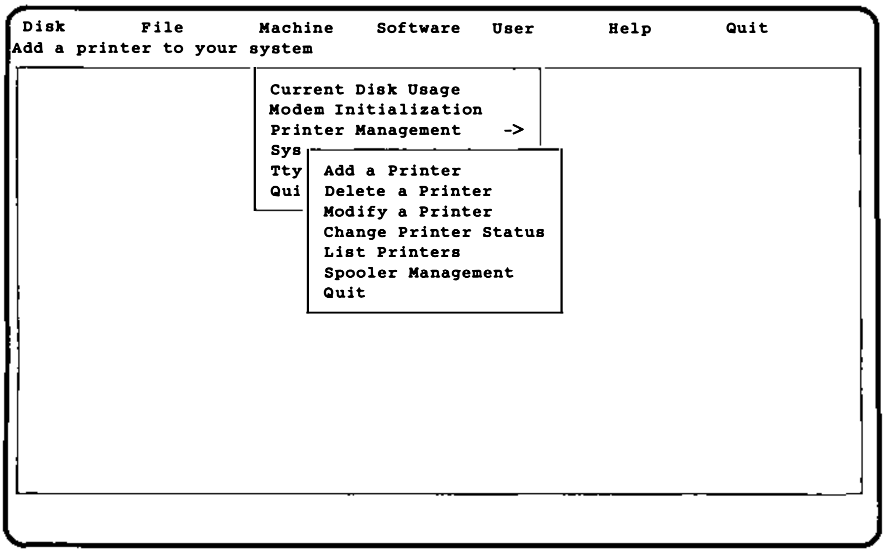
3.2.3 *The* `Machine` *Menu*

The `Machine` menu looks similar to this:



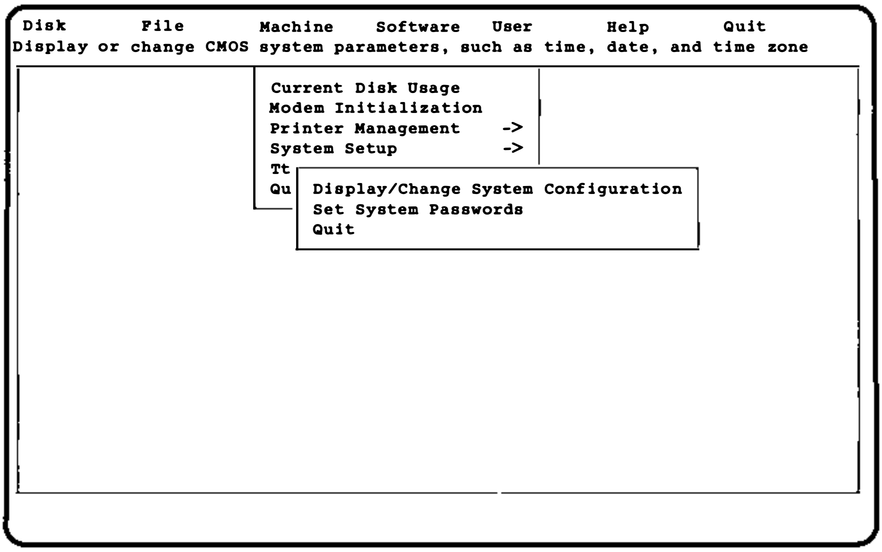
```
 Disk        File       Machine     Software    User        Help        Quit
Statistics on fixed disk(s) usage

                         Current Disk Usage
                         Modem Initialization
                         Printer Management    ->
                         System Setup          ->
                         Tty Management        ->
                         Quit
```

The `Printer Management` option displays a screen similar to this:



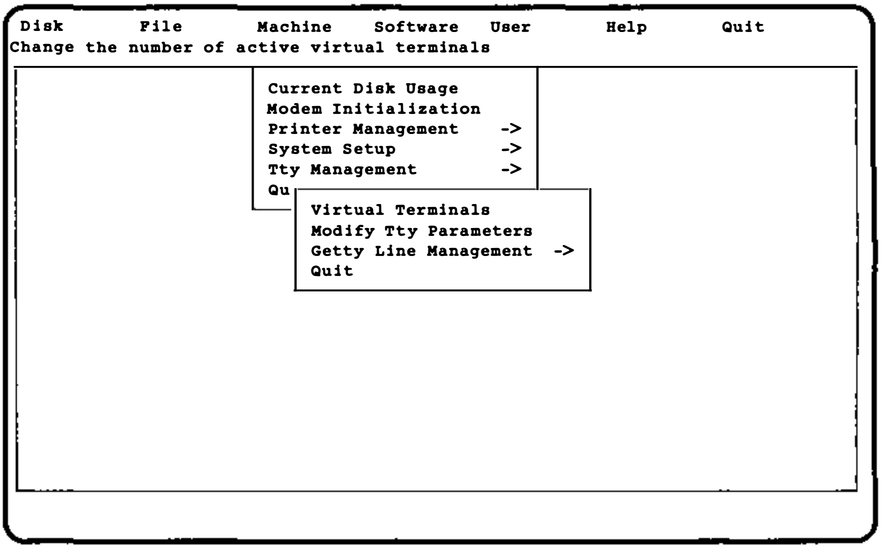
```
 Disk        File       Machine     Software    User        Help        Quit
Add a printer to your system

                         Current Disk Usage
                         Modem Initialization
                         Printer Management    ->
                         Sys
                         Tty   Add a Printer
                         Qui   Delete a Printer
                               Modify a Printer
                               Change Printer Status
                               List Printers
                               Spooler Management
                               Quit
```

The `System Setup` option displays a screen similar to this:



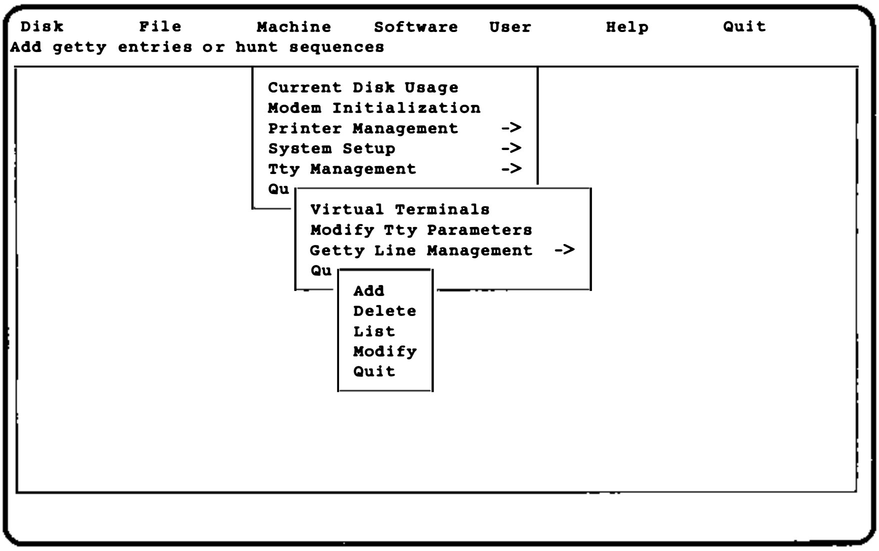
```
 Disk        File        Machine     Software   User        Help        Quit
Display or change CMOS system parameters, such as time, date, and time zone

                        Current Disk Usage
                        Modem Initialization
                        Printer Management     ->
                        System Setup           ->
                        Tt
                        Qu  Display/Change System Configuration
                            Set System Passwords
                            Quit
```

The `Tty Management` option displays a screen similar to this:



```
 Disk        File        Machine     Software   User        Help        Quit
Change the number of active virtual terminals

                        Current Disk Usage
                        Modem Initialization
                        Printer Management     ->
                        System Setup           ->
                        Tty Management         ->
                        Qu
                            Virtual Terminals
                            Modify Tty Parameters
                            Getty Line Management  ->
                            Quit
```

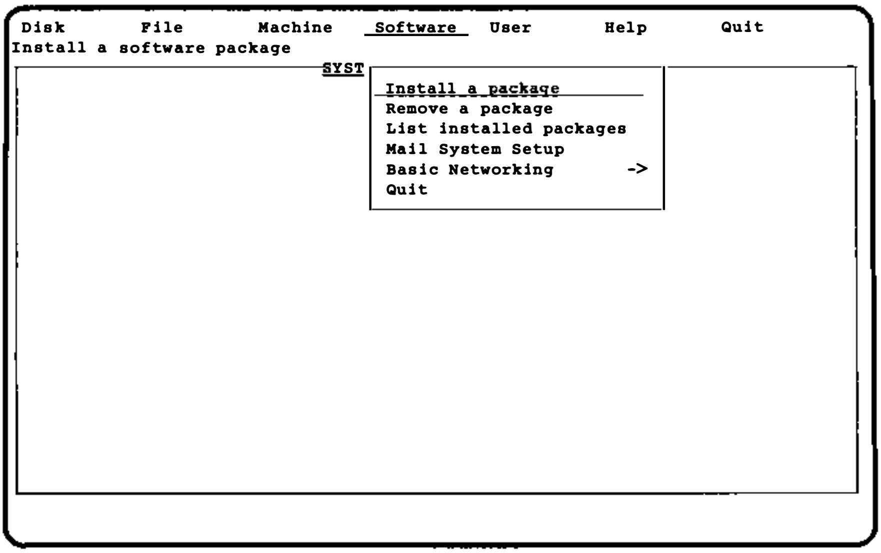
The `Getty Line Management` menu displays a screen similar to this:



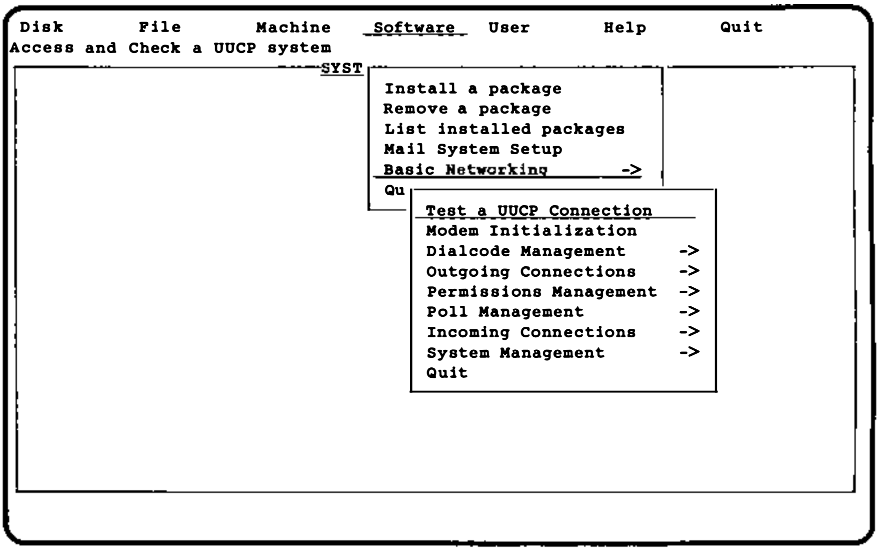
```
 Disk        File       Machine    Software   User       Help       Quit
Add getty entries or hunt sequences

                         Current Disk Usage
                         Modem Initialization
                         Printer Management    ->
                         System Setup          ->
                         Tty Management        ->
                         Qu
                             Virtual Terminals
                             Modify Tty Parameters
                             Getty Line Management  ->
                             Qu
                                 Add
                                 Delete
                                 List
                                 Modify
                                 Quit
```

3.2.4 *The* `Software` *Menu*

The `Software` menu looks similar to this:

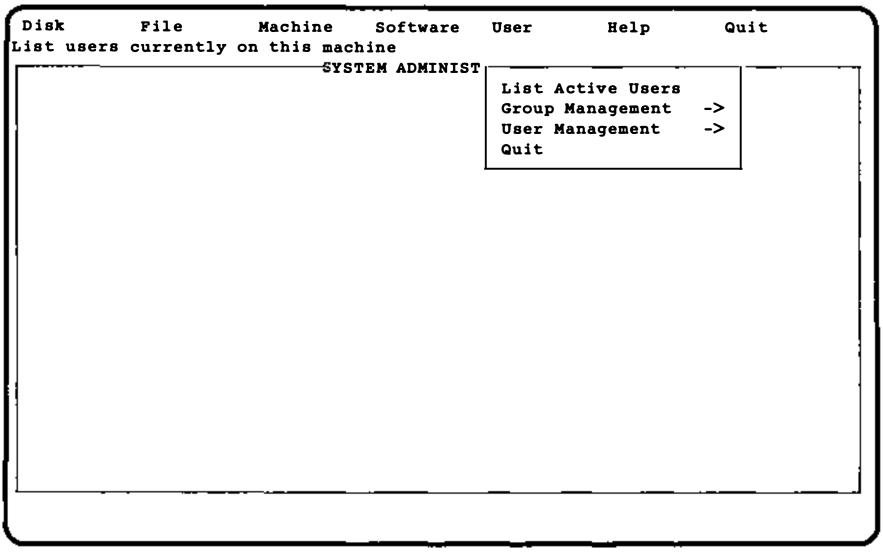

```
Disk        File        Machine     Software    User        Help        Quit
Install a software package
                                  SYST
                                        Install a package
                                        Remove a package
                                        List installed packages
                                        Mail System Setup
                                        Basic Networking        ->
                                        Quit
```

The `Basic Networking` menu looks similar to this:


```
Disk        File        Machine     Software    User        Help        Quit
Access and Check a UUCP system
                                  SYST
                                        Install a package
                                        Remove a package
                                        List installed packages
                                        Mail System Setup
                                        Basic Networking        ->
                                        Qu
                                            Test a UUCP Connection
                                            Modem Initialization
                                            Dialcode Management       ->
                                            Outgoing Connections      ->
                                            Permissions Management    ->
                                            Poll Management           ->
                                            Incoming Connections      ->
                                            System Management         ->
                                            Quit
```

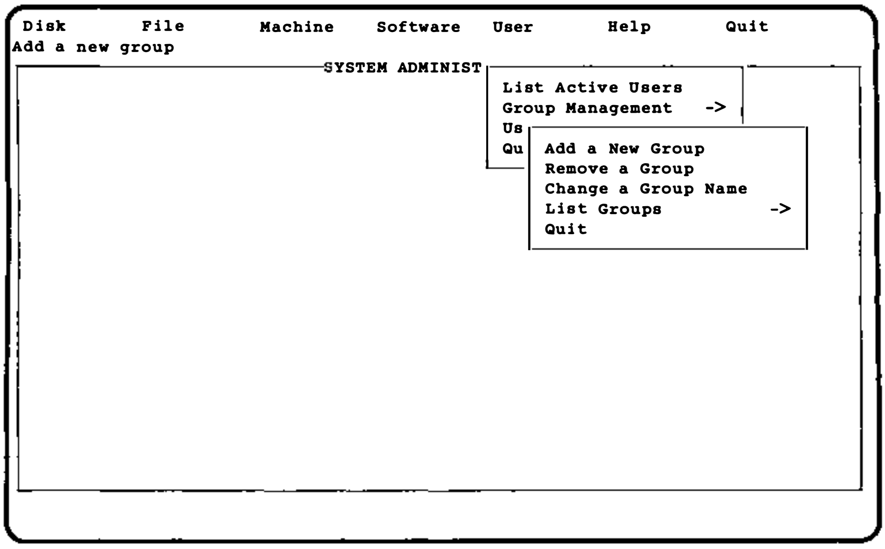
3.2.5 The `User` *Menu*

The `User` menu looks similar to this:



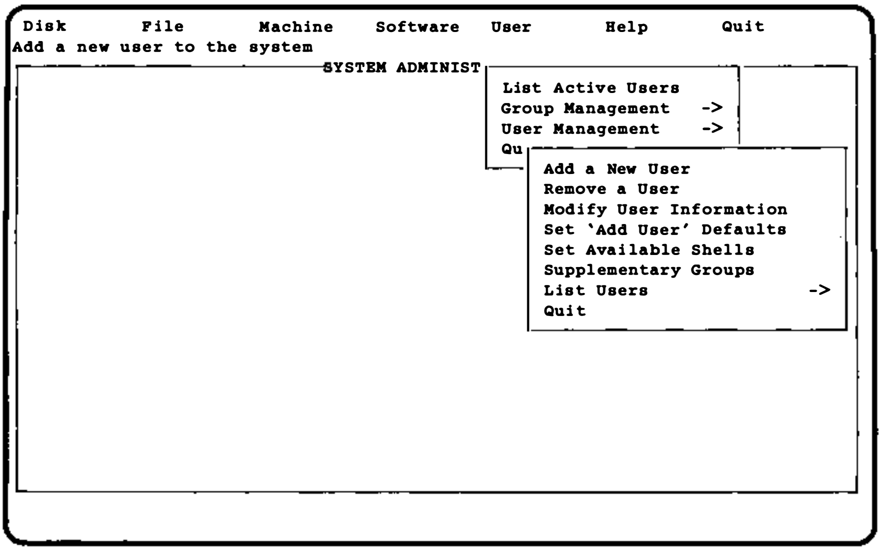
```
 Disk       File       Machine    Software   User       Help       Quit
List users currently on this machine
                                    SYSTEM ADMINIST
                                                    List Active Users
                                                    Group Management   ->
                                                    User Management    ->
                                                    Quit
```

The `Group Management` menu looks similar to this:



```
 Disk       File       Machine    Software   User       Help       Quit
Add a new group
                                    SYSTEM ADMINIST
                                                    List Active Users
                                                    Group Management   ->
                                                    Us
                                                    Qu  Add a New Group
                                                        Remove a Group
                                                        Change a Group Name
                                                        List Groups          ->
                                                        Quit
```

The `User Management` menu looks similar to this:



```
 Disk        File        Machine     Software    User        Help        Quit
Add a new user to the system
┌───────────────────────────SYSTEM ADMINIST┌──────────────────────────┐────────┐
│                                          │ List Active Users        │        │
│                                          │ Group Management    ->   │        │
│                                          │ User Management     ->   │        │
│                                          │ Qu┌─────────────────────────────┐  │
│                                          └───│ Add a New User              │  │
│                                              │ Remove a User               │  │
│                                              │ Modify User Information     │  │
│                                              │ Set 'Add User' Defaults     │  │
│                                              │ Set Available Shells        │  │
│                                              │ Supplementary Groups        │  │
│                                              │ List Users               -> │  │
│                                              │ Quit                        │  │
│                                              └─────────────────────────────┘  │
│                                                                               │
└───────────────────────────────────────────────────────────────────────────────┘
```

4. CREATING AND USING INTERACTIVE UNIX SYSTEM FILE SYSTEMS

This section describes the procedures for creating and maintaining file systems from the command line (without using `sysadm`). (For file systems other than S51K and S52K, refer to section 5.6 of that particular file system type.) For basic information on file system structure, file system and device naming conventions, disk partitioning, and using the `sysadm` commands to create and maintain INTERACTIVE UNIX System file systems, refer to sections 9 and 11 of "System Administration for New Users of the INTERACTIVE UNIX Operating System."

4.1 Creating a File System and Making It Available

After formatting a disk, you can define a file system on it using the `mkfs` command. (Note that you must have the 2 Kilobyte File System Utility Package installed to make a 2K file system.)

The `mkfs` command is used to create all file systems. One of its optional arguments is the rotational gap. For this computer, the gap should always be the default value, 2, which puts the blocks in ascending order. Thus, new files are more likely to be in sequence and are read faster. Because of this ordering, another optional argument to `mkfs` (cylinder size) is unimportant since you get the same order in all cases.

4.2 Using `mkfs`

The `mkfs` command has two formats:

```
mkfs special blocks[:i-nodes] [gap blocks/cyl] [-b blocksize]

mkfs special prototype [gap blocks/cyl] [-b blocksize]
```

Notice that the file system is not given a name in either format; it is identified by the file name of the special device file on which it will reside. The special device file, traditionally located in the directory `/dev`, is tied to the identifying controller and unit numbers (major and minor, respectively) for the physical device.

In the first format, the only other information that must be furnished on the `mkfs` command line is the number of 512-byte blocks the file system is to occupy. The second format lets you include that information in a prototype file that can also define a directory and file structure for the new file system, and it even allows for reading in the contents of files from an existing file system.

Both formats let you specify information about the interrecord gap and the blocks per cylinder. If this information is not given on the command line, default values are used. The recommendations depend on the logical block size of the file system (see the discussion of the `-b` option at the end of this section). The recommended values are different from the defaults used by the command. In the first `mkfs` format, even though the number of blocks in the file is required, the number of inodes may be omitted. If the number of inodes is omitted, the command uses a default value of one inode for every four logical storage blocks.

If you use the first format of `mkfs`, the file system is created with a single directory. If you use a prototype file, as noted above, it can include information that causes the command to build and initialize a directory and file structure for the file system. The format of a prototype file is described in *mkfs*(1M).

The final option to `mkfs` lets you specify the logical block size to be used for the file system. By default, the file system has a logical block size of 1024 bytes. (With the `-b` option, you can specify a logical block size of 512 bytes, 1024 bytes, or 2048 bytes.)

4.2.1 Choosing Logical Block Size

Logical block size is the size of the blocks the INTERACTIVE UNIX System kernel uses to read or write files. The logical block size is usually different from the physical block size, which is the size of the smallest block that the disk controller can read or write, usually 512 bytes.

An administrator who uses the `mkfs` command to make a file system may specify the logical block size of the file system. By default, the logical block size is 1024 bytes (1K). All file systems created during system installation are 1K file systems. Besides 1K file systems, the INTERACTIVE UNIX System also supports 512-byte file systems and 2048-byte (2K) file systems. To use a 2K file system, you must install the 2 Kilobyte File System Utility Package optional subset.

To choose a reasonable logical block size for your system, you must consider performance and space. Since the 1K file systems use the INTERACTIVE UNIX Operating System's Fast File System software, it is almost always the best choice for logical block size.

4.3 Creating a File System on a Diskette

You can create your own file system on a diskette by using the `mkfs` and `labelit` commands. You will actually be specifying the file system that you want on the diskette and then mounting the file system as a directory under the INTERACTIVE UNIX System. Refer to *volcopy*(1M) for additional information on `labelit`.

Creating a file system on a diskette can be very useful because the file systems are portable and it saves room on the fixed disk. The maximum size of a file system that can be created on a diskette is 720 blocks (512-byte blocks) for a 360 KB diskette, 2400 blocks (512-byte blocks) for a 1.2 MB diskette, and 2880 blocks for a 1.44 MB (high density, 3 ½ inch) diskette.

The following steps are used to create and identify a file system on a diskette.

1. Log in as `root` to assure that you have the proper read/write permissions.

2. Insert a formatted diskette into the diskette drive. Type the format command lines appropriate for the type of diskette you are using:

   If you have a 360 KB diskette, type:

   ```
   # /bin/format /dev/rdsk/f0d9dt
   ```

   and proceed to the next step.

   If you have a 1.2 MB diskette, type:

   ```
   # /bin/format /dev/rdsk/f0q15dt
   ```

   and skip to step 4. If you have a 1.44 MB diskette, type:

   ```
   # /bin/format /dev/rdsk/f0q18dt
   ```

   and skip to step 5.

3. For a 360 KB diskette, make a file system of 720 blocks and 160 inodes using the following:

   ```
   # /etc/mkfs /dev/dsk/f0d9dt 720:160 1 18
   ```

   The rotational gap is 1 and the blocks per cylinder is 18. A 360 KB diskette has 18 512-byte blocks per cylinder.

4. If you have a 1.2 MB diskette, make a file system of 2400 blocks and 592 inodes using the following:

```
# /etc/mkfs /dev/dsk/f0q15dt 2400:592 1 30
```

The rotational gap is 1 and the blocks per cylinder is 30. The 1.2 MB diskette has 30 512-byte blocks per cylinder. Skip to step 6.

5. If you have a 1.44 MB diskette, make a file system of 2880 blocks and 600 inodes using the following:

```
# /etc/mkfs /dev/dsk/f0q18dt 2880:600 1 36
```

The rotational gap is 1 and the blocks per cylinder is 36. A 1.44 MB diskette has 36 512-byte blocks per cylinder.

6. Assume that for the rest of the example you are using a 1.2 MB diskette. Regardless of the size of your diskette, your screen will look similar this:

```
# /etc/mkfs /dev/dsk/f0q15dt 2400:592 1 30

Mkfs: /dev/dsk/f0q15dt?
(DEL if wrong)
bytes per logical block = 1024
total logical blocks = 1200
total inodes = 592
gap (physical blocks) = 1
cylinder size (physical blocks) = 30
```

If the command output in this screen is not what you want, press DEL to cancel the command.

7. Label the diskette file system using the `labelit` command. For this example, assume the file system will be called `memo`. The volume name will be `memo2.0`. Type:

```
# /etc/labelit /dev/dsk/f0q15d memo memo2.0
```

Your screen will look similar to this:

```
/etc/labelit /dev/dsk/f0q15d memo memo2.0

Current fsname: , Current volname:  Blocks: 2400, Inodes: 592
FS Units: 1Kb, Date last modified: Thu Sep 10 13:24:03 1987
NEW fsname = memo, NEW volname = memo2.0 --Del if wrong!!
```

8. File systems are usually mounted in `root` (/) as directories. Make a directory in the `root` directory with the same name as the file system you are mounting:

```
# mkdir /memo
```

You must be `root` to `mount` or `umount` an INTERACTIVE UNIX Operating System file system.

9. Mount the file system as follows:

   ```
   # /etc/mount /dev/dsk/f0q15d /memo
   ```

   The file system /memo is now associated with a directory in the root file system. As long as the memo file system is mounted on /memo, you can create and modify files on it as if it were an extension to the fixed disk.

   A directory lost+found should be created on the file system for use by fsck.

Mounting a file system at a directory that does not match the file system name produces a warning message defining what has been mounted. For example, to mount /dev/dsk/f0q15dt (file system name is memo) as directory /mnt, enter the following command line:

```
# /etc/mount /dev/dsk/f0q15dt /mnt
```

The following warning message will be displayed:

```
/etc/mount: warning: <memo> mounted as </mnt>
```

4.4 Unmounting a File System

When you have finished using the file system, you should unmount it. This is done with the umount command. All files in the file system to be unmounted must be closed, and you must change to a directory not in this file system. For example, if your current directory (pwd) is in the file system you want to unmount, you must change out of the file system before executing the umount command. Otherwise, you will get the following message:

```
/etc/umount:device busy
```

To unmount a file system from a 1.2 MB diskette, type the following:

```
# /etc/umount /dev/dsk/f0q15dt
```

If the file system is unmounted cleanly, there will be no need to run fsck next time it is mounted. If it does not unmount cleanly, the next attempt to mount it will produce the following error message:

```
mount: possibly damaged file system
```

If this should happen, you can do one of two things:

- Run fsck on the file system and mount it again by typing:

  ```
  # /etc/fsck /dev/dsk/f0q15dt
  ```

- Mount the file system with read permission only as follows:

```
# /etc/mount /dev/dsk/f0q15dt /memo -r
```

4.5 root File System Free Space

A predetermined and finite amount of disk space is allocated for the `root` file system. The unoccupied disk space within this area, called free space, allows for additional and temporary files and often serves as a scratch pad for certain system programs. System administration and other types of programs require `root` file system free space to run. It is recommended that you try to avoid using all the space in the `root` file system. If you should run out of space in `root`, the following message is displayed:

```
no space on Fixed Disk Device 0x1
```

If you see this message, you should manually remove the files you do not need from the `root` file system. Since the system creates the file `/etc/mnttab` during start-up time, it is recommended that you save at least 10 free blocks in the `root` file system before shutting down the machine. The `df` command can be used to find out how many free blocks are in your file systems. Refer to *df*(1M) for more information.

5. FILE SYSTEM MAINTENANCE

When the INTERACTIVE UNIX Operating System is first installed and initialized, a `root` (`/`) file system is automatically created. Depending on the size of your fixed disk, one or more file systems, named `/usr`, `/usr2`, `/usr3` and so on, may also be created. See section 4.4 of the "INTERACTIVE UNIX Operating System Installation Instructions" for further details.

Refer to sections 10 and 11 of "System Administration for New Users of the INTERACTIVE UNIX Operating System" for information about using the `sysadm` command to do the following:

- Format a diskette
- Copy files to a diskette
- Create a file system on a diskette
- Mount a file system
- Unmount a file system
- Check and repair a file system

5.1 Checking and Repairing a File System

Every time a file is modified, the INTERACTIVE UNIX Operating System does a series of file system updates. These updates, when written to the disk, produce a consistent file system. The components of a file system are:

super-block	The super-block defines the internal structure and size of a file system. There is one super-block for each file system.
inodes	An information node (or inode) is the internal definition of a file or directory. An inode contains information about the type of file, the number of directory entries linked to the file, a list of blocks claimed by the file, and the size of the file.
data blocks	A data block can contain either directory entries or file data. Each directory entry consists of a file name and an inode number. Each data block contains 1024 bytes.

indirect blocks	Indirect blocks are needed to reference the data blocks of large files (more than 10 blocks long). There are three types of indirect blocks: single-indirect, double-indirect, and triple-indirect.
first free-list block	The free-list blocks are lists of all the blocks not allocated to the super-block, inodes, or existing files. The super-block points to the first free-list block.

5.1.1 File System Reliability Features

The INTERACTIVE UNIX System is always checking to see if your file systems are in working order. Your system has several reliability features built in. For example:

- When a file is written to the fixed disk, its inode and blocks are written in an order that ensures maximum reliability. This is known as ordered writes.
- System buffers are periodically written to the fixed disk to keep the file contents up to date. This is known as automatic update.
- If the file system becomes corrupted, you will be required to run the `fsck` program to clean up the file system before mounting it. This ensures the reliability of all computer-mounted file systems.

5.1.2 The `fsck` Program

The `fsck` program is a file system check-and-repair program that uses information that is stored in the file system to check for inconsistencies. When the INTERACTIVE UNIX Operating System is booted, your computer runs a consistency check on the status of the `root` file system. If a potential problem exists, or if the information it checks does not match, the `fsck` program automatically detects the problem or inconsistency and attempts to repair it. Because `fsck` runs automatically on the `root` file system when the system is booted, you should not have to run `fsck` manually for the `root` file system. The `fsck` command is usually located in `/etc`.

The `fsck` program can also be run manually to check diskettes that have INTERACTIVE UNIX System file systems on them; or if you suspect something is wrong with your file system, you may want to check it. This should only be attempted by expert users.

Refer to section 11.5 of "System Administration for New Users of the INTERACTIVE UNIX Operating System" for information about using the `Check a Diskette File System` option from the `File System Management` menu from the `Diskette Management` menu under the `Disk` menu to run `fsck` on damaged diskettes and using the `Check a Fixed Disk File System` option on the `Fixed Disk Management` menu under the `Disk` menu to run `fsck` on a damaged file system on a fixed disk.

If the system experiences a power failure, it is usually possible to recover from it without suffering extensive file system damage. However, if the system experiences a hardware failure, the file system may be damaged beyond repair.

If file system damage is relatively minor, `fsck` will automatically repair many of the problems without assistance from you. In some cases the system asks you to confirm the action it plans to take. If `fsck` cannot repair the problem, you may have to reinstall a backup version of the file system.

After the initial setup, `fsck` performs successive phases of tests over the file system, including cleanup and checking blocks and sizes, path names, connectivity, reference counts, and the free-block list (possibly rebuilding it).

When an inconsistency is detected, `fsck` reports the error condition to the user. If a response is required, `fsck` prints a prompt message and waits for a response. The following paragraphs explain the meaning of an error condition, the possible responses, and the related error conditions.

The error conditions are organized by the "phase" of the `fsck` program in which they can occur.

For a list and explanation of the error messages you could encounter when using `fsck`, refer to section 5.4.

5.1.2.1 Phase 1: Check Blocks and Sizes. This phase concerns itself with the inode list. Activities include checking inode types, setting up the zero-link-count table, examining inode block numbers for bad or duplicate blocks, checking inode size, and checking inode format.

5.1.2.2 Phase 1B: Rescan for More DUPS. When a duplicate block is found in the file system, the file system is rescanned to find the inode that previously claimed that block.

5.1.2.3 Phase 2: Check Path Names. This phase concerns itself with removing directory entries pointing to error-conditioned inodes from Phase 1 and Phase 1B. Checks are run for `root` inode mode and status, directory inode pointers in range, and directory entries pointing to bad inodes.

5.1.2.4 Phase 3: Check Connectivity. This phase concerns itself with the directory connectivity seen in Phase 2. This part lists error conditions resulting from unreferenced directories and missing or full `lost+found` directories.

5.1.2.5 Phase 4: Check Reference Counts. This phase concerns itself with the link count information seen in Phase 2 and Phase 3. This part lists error conditions resulting from unreferenced files; missing or full `lost+found` directories; incorrect link count for files, directories, and special files; unreferenced files and directories; bad and duplicate blocks in files and directories; and incorrect total free inode counts.

5.1.2.6 Phase 5: Check Free List. This phase concerns itself with the free-block list. This part lists error conditions resulting from bad blocks in the free-block list, bad free-block count, duplicate blocks in the free-block list, unused blocks from the file system not in the free-block list, and an incorrect total free-block count.

5.1.2.7 Phase 6: Salvage Free List. This phase concerns itself with the free-block list reconstruction. This part lists error conditions resulting from the blocks-to-skip and blocks-per-cylinder values.

5.1.2.8 Cleanup. Once a file system has been checked, a few cleanup functions are performed. This lists the following advisory messages about the file system and modifies status of the file system:

```
***** FILE SYSTEM STATE SET TO OKAY *****
```

A flag in the super-block will be set to indicate that the file system is not corrupted and can be mounted.

```
X files Y blocks Z free
```

This advisory message indicates that the file system checked contained *X* files using *Y* blocks leaving *Z* blocks free in the file system.

```
***** FSCK and the ROOT FILE SYSTEM *****
```

`root` is the only file system that can (and must) be checked while mounted. Automated mechanisms are provided for checking the `root` file system. These mechanisms handle a dirty `root` when booting and periodic checks during shutdown. You can also force a check on shutdown. These mechanisms hide the messages from `fsck`. If they were not hidden, you would see the error message described next.

```
***** ROOT FILE SYSTEM WAS MODIFIED *****
```

This advisory message indicates that the `root` file system was modified by `fsck`. If a system reboot is necessary, `fsck` with the `-b` option forces an automatic reboot and prints the following message:

```
**** SYSTEM WILL REBOOT AUTOMATICALLY ****
```

If you decide not to use the automated mechanisms, the `-b` option is not used, and a system reboot is necessary; press **RESET**.

The automated procedures establish the proper environment (no processes fiddling with files) for checking `root`.

5.1.3 Running `fsck` *Manually*

To run the `fsck` program manually, the file system must be unmounted (with the exception of the `root` file system). The legal `fsck` options are `-b`, `-f`, `-y`, `-n`, `-s`, `-S`, `-t`, `-q`, and `-D`. The `-y` option is recommended for `fsck`. This option answers yes to all questions prompted by `fsck` and requires no user intervention. Use the following command line for `fsck`:

```
# /etc/fsck -y special
```

Your screen will look similar to this:

```
/dev/dsk/0s0
File System:   Volume:

**Phase1 - Check Blocks and Sizes
POSSIBLE FILE SIZE ERROR I=321

POSSIBLE FILE SIZE ERROR I=394

**Phase 2 - Check Pathnames
**Phase 3 - Check Connectivity
**Phase 4 - Check Reference Counts
**Phase 5 - Check Free List
 411 files 4394 blocks 8880 free
```

Refer to *fsck*(1M) for additional information. If you attempt to use the `fsck` command on a mounted file system other than the `root` file system, the following message is displayed:

```
/dev/dsk/?? is a mounted file system, ignored.

?? is the special device name.
```

5.2 Bad Block Handling

The requirements of bad block handling fall into six categories:

- Dynamic handling of bad blocks
- Maintaining a bad block mapping table
- Detecting bad blocks
- Detecting unreadable blocks
- Reporting bad blocks
- Remapping bad blocks

5.2.1 Dynamic Handling of Bad Blocks

The basic requirement for the bad block handling feature is that it must be done dynamically, without user intervention. Dynamic handling provides immediate attention to the problem and thus minimizes data loss. It also avoids errors that may be introduced by the user. Our current implementation reports problems to the console as they are found without retaining the messages in a log.

5.2.2 Maintaining a Bad Block Mapping Table

The bad block mapping table is created and stored on the fixed disk when the disk is first formatted. It consists of a bad block list of alternate blocks commonly called the "alternate sector list or surrogate images." These two lists are in a one-to-one correspondence.

The bad block list is used to record the address, on disk, of the blocks that are bad. The alternate sector list is used to record the address, on disk, of all the reserved sectors to be used as alternates for bad blocks.

5.2.3 Detecting Bad Blocks

The bad block handling feature should be able to detect two different types of problems:

- Marginal blocks
- Unreadable blocks

A marginal block is a block that is readable, but with some difficulty. That is, the fixed disk controller's Error Correction Code (ECC) algorithm has to be used to successfully read the block.

5.2.4 Detecting Unreadable Blocks

An unreadable block is a harder problem to solve. There are two possible solutions. One method deals with the possible reconstruction of data to minimize data loss. The other simply accepts that the data is lost.

Reconstruction of data requires that a thorough and extensive analysis of the block in question is done before any kind or form of data repair can be attempted.

While this method offers higher data conservation, its design and implementation requires a considerable amount of time and effort. Implementation of this method did not occur at this time; however, it is being considered as a future extension to the bad block handling feature.

It should be noted that by having an implementation in place that detects and takes care of marginal blocks, the incidence of potentially unreadable blocks is greatly reduced.

5.2.5 Reporting Bad Blocks

The system displays an error message when bad blocks are encountered. The following message may appear on the console:

```
*** DEVICE ERROR: Data Address Mark not found ***
*** Controller 0 (Primary AT Hard Disk), Disk Drive 0,
*** Absolute Sector # 168328
***
```

The system reports that absolute sector number 168328 on disk driver 0, controller 0, is bad. Note that the driver string, `Primary AT Hard Disk`, may be different depending on the controller configuration.

5.2.6 Remapping Bad Blocks

Bad blocks may be remapped into the reserved alternate sectors using the `-A` option of the `mkpart` command. (Refer to *mkpart*(1) for more information.) For example, to remap the bad sector in the above example, type in:

```
mkpart -A 168328 disk_c0t0
```

The `mkpart` command can also remap multiple bad blocks on a single command line. Thus, to remap bad sectors 100000 and 200000, type in:

```
mkpart -A 100000 -A 200000 disk_c0t0
```

5.3 Recovery of the INTERACTIVE UNIX Operating System

If the message `/unix is missing or corrupted.` appears when booting your computer or when the file system becomes corrupted to the point where the system is inoperable, you will need to replace `/unix` with the default `/unix`. Try the following:

1. Insert the *Boot* diskette into the diskette drive and press RESET.
2. Remove the *Boot* diskette and insert the *Install* diskette into the diskette drive when prompted to do so.
3. At this point, you may or may not be asked to authenticate your software. If you are asked, type in the appropriate serial number and authorization code. In any case, *after* the system asks you to choose your keyboard nationality, press CTRL \ to break out of the installation procedure.
4. Run `fsck` on the `root` file system on your fixed disk (from the `root` prompt) by typing:

   ```
   # /etc/fsck /dev/dsk/0s1
   ```

   The `fsck` command will either run with no errors or request action from the user on repairing the file system. Most of the time answering `yes` to the questions asked by `fsck` will be sufficient, but be aware this could remove some files.
5. Mount the fixed disk (device `0s1`) by typing:

   ```
   # mount /dev/dsk/0s1 /mnt
   ```
6. Type:

   ```
   # init switchroot
   ```

   to change `root` to the fixed disk.
7. Type:

   ```
   # exit
   ```

   and ignore any messages that appear at this point.
8. Remove the *Install* diskette and insert the *Boot* diskette.
9. Mount the diskette by typing:

   ```
   # mount /dev/dsk/f0q15d /mnt
   ```
10. Copy the kernel (`/unix`) from the *Boot* diskette to the fixed disk by typing:

```
# cp /mnt/unix /unix
```

11. Unmount the *Boot* diskette by typing:

```
# umount /mnt
```

12. Remove the *Boot* diskette and type:

```
# sync; sync; uadmin 2 0
```

    to shut down your system cleanly. Press any key to reboot when prompted.

☛ When you get the INTERACTIVE UNIX System prompt, make sure you are logged in as `root`. You should then reinstall all drivers previously installed using the `kconfig` command. Refer to *kconfig*(1) for more information.

5.4 `fsck` Error Messages

This section describes the error messages you might receive when using `fsck`.

5.4.1 Initialization

Before a file system check can be performed, certain tables have to be set up and certain files opened. This section describes the opening of files and the initialization of tables. Error conditions resulting from command line options, memory requests, opening of files, status of files, file system size checks, and creation of the scratch file are listed below. The `fsck` program terminates on initialization errors.

5.4.2 Error Messages

`Bad -t option Illegal scratch file` <*device name*>
: The `-t` option is not followed by a file name. The `fsck` program terminates on this error condition.

`Invalid -s argument, defaults assumed`
: The `-s` option is not suffixed by 3, 4, or blocks-per-cylinder:blocks-to-skip. The `fsck` program assumes a default of 400 blocks-per-cylinder and 7 blocks-to-skip.

`Incompatible options: -n and -q`
: It is not possible to remove FIFOs without modifying the file system. The `fsck` program terminates on this error condition.

`Incompatible options: -n and -s`

It is not possible to salvage the free-block list without modifying the file system. The `fsck` program terminates on this error condition.

`Cannot fstat standard input`

The attempt to `fstat` standard input failed. This error condition indicates a serious problem that may require additional assistance. The `fsck` program terminates on this error condition.

`Can't get memory`

The request for memory for virtual memory tables failed. This error condition indicates a serious problem that may require additional assistance. The `fsck` program terminates on this error condition.

`Cannot open checklist file: F`

The default file system *checklist* file *F* (usually `/etc/checklist`) cannot be opened for reading. The `fsck` program terminates on this error condition. Check access modes of *F*.

`Cannot stat root`

The request for statistics about the `root` directory failed. This error condition indicates a serious problem that may require additional assistance. The `fsck` program terminates on this error condition.

`Cannot stat F`

The request for statistics about the file system *F* failed. The `fsck` program ignores this file system and continues checking the next file system given. Check access modes of *F*.

`F is not a block or character device`

The `fsck` program has been given a regular file name by mistake. It ignores this argument and continues checking the next file system given. Check the file type of *F*.

`Can't open F`

The file system *F* cannot be opened for reading. The `fsck` program ignores this and continues checking the next file system given. Check the access modes of *F*.

`Size check: fsize` *X* `isize` *Y*

More blocks are used for the inode list *Y* than there are blocks in the file system *X*, or there are more than 65,535 inodes in the file system. The `fsck` program ignores this file system and continues checking the next file system given.

`Can't create F`

The request to create a scratch file *F* failed. The `fsck` program ignores this file system and continues checking the next file system given. Check the access modes of *F*.

`CAN NOT SEEK: BLK` *B* `(CONTINUE)`

The request for moving to a specified block number *B* in the file system failed. The occurrence of this error condition indicates a serious problem that may require additional assistance.

Possible responses to the `CONTINUE` prompt are:

`YES` Attempt to continue to run file system check. Often, however, the problem persists. This error condition does not allow a complete check of the file system. A second run of `fsck` should be made to recheck this file system. If the block was part of the virtual memory buffer cache, `fsck` will terminate with the message `Fatal I/O error`.

`NO` Terminate program.

`CAN NOT READ: BLK` *B* `(CONTINUE)`

The request for reading a specified block number *B* in the file system failed. The occurrence of this error condition indicates a serious problem that may require additional assistance.

Possible responses to the `CONTINUE` prompt are:

`YES` Attempt to continue to run file system check. Often, however, the problem persists. This error condition does not allow a complete check of the file system. A second run of `fsck` should be made to recheck this file system. If block was part of the virtual memory buffer cache, `fsck` will terminate with the message `Fatal I/O error`.

`NO` Terminate program.

`CAN NOT WRITE: BLK` *B* `(CONTINUE)`

The request for writing a specified block number *B* in the file system failed. The file system should not be opened for writing.

Possible responses to the `CONTINUE` prompt are:

`YES`
: Attempt to continue to run file system check. Often, however, the problem persists. This error condition does not allow a complete check of the file system. A second run of `fsck` should be made to recheck this file system. If block was part of the virtual memory buffer cache, `fsck` terminates with the message `Fatal I/O error`.

`NO`
: Terminate program.

5.4.3 Phase 1: Check Blocks and Sizes

This phase concerns itself with the inode list. This part lists error conditions resulting from checking inode types, setting up the zero-link-count table, examining inode block numbers for bad or duplicate blocks, checking inode size, and checking inode format.

`UNKNOWN FILE TYPE I=I (CLEAR)`

The mode word of the inode *I* indicates that the inode is not a special character inode, regular inode, or directory inode.

Possible responses to the `CLEAR` prompt are:

`YES` Deallocate inode *I* by zeroing its contents. This invokes the `UNALLOCATED` error condition in Phase 2 for each directory entry pointing to this inode.

`NO` Ignore this error condition.

`LINK COUNT TABLE OVERFLOW (CONTINUE)`

An internal table for `fsck` containing allocated inodes with a link count of zero has no more room.

Possible responses to the `CONTINUE` prompt are:

`YES` Continue with program. This error condition does not allow a complete check of the file system. A system run of `fsck` should be made to recheck this file system. If another allocated inode with a zero link count is found, this error condition will be repeated.

`NO` Terminate the program.

`B BAD I=I`

Inode *I* contains block number *B* with a number lower than the number of the first data block in the file system or greater than the number of the last block in the file system. This error condition may invoke the `EXCESSIVE BAD BLKS` error condition in Phase 1 if inode *I* has too many block numbers outside the file system range. This error condition invokes the `BAD/DUP` error condition in Phase 2 and Phase 4.

`EXCESSIVE BAD BLKS I=I (CONTINUE)`

There is more than a tolerable number (usually 10) of blocks claimed by other inodes.

Possible responses to the `CONTINUE` prompt are:

`YES` Ignore the rest of the blocks in this inode and continue to check using the next inode in the file system. This error condition does not allow a complete check of the file system. A second run of `fsck` should be made to recheck this file system.

`NO` Terminate the program.

`B DUP I=I`

Inode *I* contains block number *B* that is already claimed by another inode. This error condition may invoke the `EXCESSIVE DUP BLKS` error condition in Phase 1 if inode *I* has too many block numbers claimed by other inodes. This error condition invokes Phase 1B and the `BAD/DUP` error condition in Phase 2 and Phase 4.

`EXCESSIVE DUPS BLKS I=I (CONTINUE)`

There is more than a tolerable number (usually 10) of blocks claimed by other inodes.

Possible responses to the `CONTINUE` prompt are:

`YES` Ignore the rest of the blocks in this inode and continue to check using the next inode in the file system. This error condition does not allow a complete check of the file system. A second run of `fsck` should be made to recheck this file system.

`NO` Terminate the program.

`DUP TABLE OVERFLOW (CONTINUE)`

An internal table in `fsck` containing duplicate block numbers has no more room.

Possible responses to the `CONTINUE` prompt are:

`YES` Continue with program. This error condition does not allow a complete check of the file system. A second run of `fsck` should be made to recheck this file system. If another duplicate block is found, this error condition will repeat.

`NO` Terminate the program.

`POSSIBLE FILE SIZE ERROR I=I`

The inode *I* size does not match the actual number of blocks used by the inode. This is only a warning. If the `-q` option is used, this message will not print.

`DIRECTORY MISALIGNED I=I`

The size of a directory inode is not a multiple of the size of a directory entry (usually 16). This is only a warning. If the `-q` option is used, this message will not print.

`PARTIALLY ALLOCATED INODE I=I (CLEAR)`

Inode *I* is neither allocated nor unallocated.

Possible responses to the `CLEAR` prompt are:

`YES` Deallocate inode *I* by zeroing its contents.

`NO` Ignore this error condition.

5.4.4 Phase 1B: Rescan for More DUPS

When a duplicate block is found in the file system, the file system is rescanned to find the inode that previously claimed that block. This part lists the error condition when the duplicate block is found.

`B DUP I=I`

Inode *I* contains block number *B* that is already claimed by another inode. This error condition invokes the `BAD/DUP` error condition in Phase 2. Inodes with overlapping blocks may be determined by examining this error condition and the `DUP` error condition in Phase 1.

5.4.5 Phase 2: Check Path Names

This phase concerns itself with removing directory entries pointing to inodes from Phase 1 and Phase 1B that have error conditions. This part lists error conditions resulting from `root` inode mode and status, directory inode pointers in range, and directory entries pointing to bad inodes.

`ROOT INODE UNALLOCATED. TERMINATING`

The `root` inode (always inode number 2) has no allocated mode bits. The occurrence of this error condition indicates a serious problem that may require additional assistance. The program stops.

`ROOT INODE NOT DIRECTORY (FIX)`

The `root` inode (usually inode number 2) is not directory inode type.

Possible responses to the `FIX` prompt are:

`YES`	Replace the `root` inode type to be a directory. If the `root` inode data blocks are not directory blocks, a large number of error conditions will be produced.
`NO`	Terminate the program.

`DUPS/BAD IN ROOT INODE (CONTINUE)`

Phase 1 or Phase 1B has found duplicate blocks or bad blocks in the `root` inode (usually inode number 2) for the file system.

Possible responses to the `CONTINUE` prompt are:

`YES`	Ignore `DUPS/BAD` error condition in `root` inode and attempt to continue to run the file system check. If `root` inode is not correct, then this may result in a large number of other error conditions.
`NO`	Terminate the program.

`I OUT OF RANGE I=I NAME=F (REMOVE)`

A directory entry *F* has an inode number *I* that is greater than the end of the inode list.

Possible responses to the `REMOVE` prompt are:

`YES`	The directory entry *F* is removed.
`NO`	Ignore this error condition.

`UNALLOCATED I=I OWNER=O MODE=M SIZE=S MTIME=T NAME=F (REMOVE)`

A directory entry *F* has an inode *I* without allocate mode bits. The owner *0*, mode *M*, size *S*, modify time *T*, and file name *F* are printed. If the file system is not mounted and the `-n` option is not specified, the entry will be removed automatically if the inode it points to is size 0.

Possible responses to the `REMOVE` prompt are:

`YES`	The directory entry *F* is removed.
`NO`	Ignore this error condition.

`DUP/BAD I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F (REMOVE)`

Phase 1 or Phase 1B has found duplicate blocks or bad blocks associated with directory entry *F*, inode *I*. The owner *O*, mode *M*, size *S*, modify time *T*, and file name *F* are printed.

Possible responses to the `REMOVE` prompt are:

`YES`	The directory entry *F* is removed.
`NO`	Ignore this error condition.

`DUP/BAD I=I OWNER=O MODE=M SIZE=S MTIME=T FILE=F (REMOVE)`

Phase 1 or Phase 1B has found duplicate blocks or bad blocks associated with directory entry *F*, inode *I*. The owner *O*, mode *M*, size *S*, modify time *T*, and file name *F* are printed.

Possible responses to the `REMOVE` prompt are:

`YES`	The directory entry *F* is removed.
`NO`	Ignore this error condition.

`BAD BLK B IN DIR I=I OWNER=O MODE=M SIZE=S MTIME=T`

This message only occurs when the `-q` option is used. A bad block was found in `DIR` inode *I*. Error conditions looked for in directory blocks are nonzero padded entries, inconsistent

"." and ".." entries, and embedded slashes in the name field. This error message indicates that the user should at a later time either remove the directory inode if the entire block looks bad or change (or remove) those directory entries that look bad.

5.4.6 Phase 3: Check Connectivity

This phase concerns itself with the directory connectivity seen in Phase 2. This part lists error conditions resulting from unreferenced directories and missing or full `lost+found` directories.

`UNREF DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (RECONNECT)`

The directory inode *I* was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of directory inode *I* are printed. The `fsck` program forces the reconnection of a nonempty directory.

Possible responses to the `RECONNECT` prompt are:

`YES`	Reconnect directory inode *I* to the file system in directory for lost files (usually `lost+found`). This may invoke `lost+found` error condition in Phase 3 if there are problems connecting directory inode *I* to `lost+found`. This may also invoke `CONNECTED` error condition in Phase 3 if link was successful.
`NO`	Ignore this error condition. This invokes `UNREF` error condition in Phase 4.

`SORRY, NO lost+found DIRECTORY`

There is no `lost+found` directory in the `root` directory of the file system; `fsck` ignores the request to link a directory in `lost+found`. This invokes the `UNREF` error condition in Phase 4. Check access modes of `lost+found`.

`SORRY, NO SPACE IN lost+found DIRECTORY`

There is no space to add another entry to the `lost+found` directory in the `root` directory of the file system; `fsck` ignores the request to link a directory in `lost+found`. This invokes the `UNREF` error condition in Phase 4. Clean out unnecessary entries in `lost+found` or make `lost+found` larger.

`DIR I=I1 CONNECTED, PARENT WAS I=I2`

This is an advisory message indicating a directory inode *I1* was successfully connected to the `lost+found` directory. The parent inode *I2* of the directory inode *I1* is replaced by the inode number of the `lost+found` directory.

5.4.7 Phase 4: Check Reference Counts

This phase concerns itself with the link count information seen in Phase 2 and Phase 3. This part lists error conditions resulting from unreferenced files; a missing or full `lost+found` directory; an incorrect link count for files, directories, or special files; unreferenced files and directories; bad and duplicate blocks in files and directories; and incorrect total free-inode counts.

`UNREF FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (RECONNECT)`

Inode *I* was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. If the `-n` option is omitted and the file system is not mounted, empty files will be cleared automatically. Nonempty directories are not cleared.

Possible responses to the `RECONNECT` prompt are:

`YES`	Reconnect inode *I* to file system in the directory for lost files (usually `lost+found`). This can cause a `lost+found` error condition in Phase 4 if there are problems connecting inode *I* to `lost+found`.
`NO`	Ignore this error condition. This invokes a `CLEAR` error condition in Phase 4.

`SORRY. NO lost+found DIRECTORY`

There is no `lost+found` directory in the `root` directory of the file system; `fsck` ignores the request to link a file in `lost+found`. This invokes the `CLEAR` error condition in Phase 4. Check access modes of `lost+found`.

`SORRY. NO SPACE IN lost+found DIRECTORY`

There is no space to add another entry to the `lost+found` directory in the `root` directory of the file system; `fsck` ignores the request to link a file in `lost+found`. This invokes the `CLEAR` error condition in Phase 4. Check size and contents of `lost+found`.

`(CLEAR)`

The inode mentioned in the immediately previous error condition cannot be reconnected.

Possible responses to the `CLEAR` prompt are:

`YES`	Deallocate inode mentioned in the immediately previous error condition by zeroing its contents.
`NO`	Ignore this error condition.

`LINK COUNT FILE I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X SHOULD BE Y (ADJUST)`

The link count for inode *I*, which is a file, is *X* but should be *Y*. The owner *O*, mode *M*, size *S*, and modify time *T* are printed.

Possible responses to the `ADJUST` prompt are:

`YES`	Replace link count of file inode *I* with *Y*.
`NO`	Ignore this error condition.

`LINK COUNT DIR I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X SHOULD BE Y (ADJUST)`

The link count for inode *I*, which is a directory, is *X* but should be *Y*. The owner *O*, mode *M*, size *S*, and modify time *T* of directory inode *I* are printed.

Possible responses to the `ADJUST` prompt are:

`YES`	Replace link count of directory inode *I* with *Y*.
`NO`	Ignore this error condition.

`LINK COUNT F I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X SHOULD BE Y (ADJUST)`

The link count of *F* inode *I* is *X* but should be *Y*. The file name *F*, owner *O*, mode *M*, size *S*, and modify time *T* are printed.

Possible responses to the `ADJUST` prompt are:

`YES`	Replace link count of inode *I* with *Y*.
`NO`	Ignore this error condition.

`UNREF FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR)`

Inode *I*, which is a file, was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. If the `-n` option is omitted and the file system is not mounted, empty files will be cleared automatically. Nonempty directories are not cleared.

Possible responses to the `CLEAR` prompt are:

`YES`	Deallocate inode *I* by zeroing its contents.
`NO`	Ignore this error condition.

`UNREF DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR)`

Inode *I*, which is a directory, was not connected to a directory entry when the file system was traversed. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed. If the `-n` option is omitted and the file system is not mounted, empty files will be cleared automatically. Nonempty directories are not cleared.

Possible responses to the `CLEAR` prompt are:

`YES`	Deallocate inode *I* by zeroing its contents.
`NO`	Ignore this error condition.

`BAD/DUP FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR)`

Phase 1 or Phase 1B has found duplicate blocks or bad blocks associated with the file inode *I*. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed.

Possible responses to the `CLEAR` prompt are:

`YES`	Deallocate inode *I* by zeroing its contents.
`NO`	Ignore this error condition.

`BAD/DUP DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR)`

Phase 1 or Phase 1B has found duplicate blocks or bad blocks associated with directory inode *I*. The owner *O*, mode *M*, size *S*, and modify time *T* of inode *I* are printed.

Possible responses to the `CLEAR` prompt are:

`YES` Deallocate inode *I* by zeroing its contents.

`NO` Ignore this error condition.

`FREE INODE COUNT WRONG IN SUPERBLK (FIX)`
The actual count of the free inodes does not match the count in the super-block of the file system. If the `-q` option is specified, the count will be fixed automatically in the super-block.

Possible responses to the `FIX` prompt are:

`YES` Replace count in super-block by actual count.

`NO` Ignore this error condition.

5.4.8 Phase 5: Check Free List

This phase concerns itself with the free-block list. This part lists error conditions resulting from bad blocks in the free-block list, bad free-block count, duplicate blocks in the free-block list, unused blocks from the file system not in the free-block list, and the total free-block count incorrect.

`EXCESSIVE BAD BLKS IN FREE LIST (CONTINUE)`
The free-block list contains more than a tolerable number (usually 10) of blocks with a value less than the first data block in the file system or greater than the last block in the file system.

Possible responses to the `CONTINUE` prompt are:

`YES` Ignore the rest of the free-block list and continue execution of `fsck`. This error condition will always invoke `BAD BLKS IN FREE LIST` error condition in Phase 5.

`NO` Terminate the program.

`EXCESSIVE DUP BLKS IN FREE LIST (CONTINUE)`
The free-block list contains more than a tolerable number (usually 10) of blocks claimed by inodes or earlier parts of the free-block list.

Possible responses to the `CONTINUE` prompt are:

`YES` Ignore the rest of the free-block list and continue execution of `fsck`. This error condition will always invoke DUP BLKS IN FREE LIST error condition in Phase 5.

NO Terminate the program.

BAD FREEBLK COUNT

The count of free blocks in a free-list block is greater than 50 or less than 0. This error condition will always invoke the BAD FREE LIST condition in Phase 5.

X BAD BLKS IN FREE LIST

X blocks in the free-block list have a block number less than the first data block in the file system or greater than the last block in the file system. This error condition will always invoke the BAD FREE LIST condition in Phase 5.

X DUP BLKS IN FREE LIST

X blocks claimed by inodes or earlier parts of the free-list block were found in the free-block list. This error condition will always invoke the BAD FREE LIST condition in Phase 5.

X BLK(S) MISSING

X blocks unused by the file system were not found in the free-block list. This error condition will always invoke the BAD FREE LIST condition in Phase 5.

FREE BLK COUNT WRONG IN SUPERBLOCK (FIX)

The actual count of free blocks does not match the count in the superblock of the file system. When a file system is mounted, the free list is converted to a bit map and the blocks are removed from the free list (a small number of blocks are reserved on the free list). If the system goes down before the bit map is reconverted to the free list, then this message is displayed. Answering yes to the fix prompt will restore the blocks to the free list.

Possible responses to the FIX prompt are:

YES Replace count in super-block by actual count.

NO Ignore this error condition.

BAD FREE LIST (SALVAGE)

Phase 5 has found bad blocks in the free-block list, duplicate blocks in the free-block list, or blocks missing from the file system. If the `-q` option is specified, the free-block list will be salvaged automatically.

Possible responses to the `SALVAGE` prompt are:

`YES`	Replace actual free-block list with a new free-block list. The new free-block list will be ordered to reduce the time spent by the disk rotating into position.
`NO`	Ignore this error condition.

5.4.9 Phase 6: Salvage Free List

This phase concerns itself with the free-block list reconstruction. This part lists error conditions resulting from the blocks-to-skip and blocks-per-cylinder values.

`Default free-block list spacing assumed`
This is an advisory message indicating that blocks-to-skip (gap size) is greater than blocks-per-cylinder, blocks-to-skip is less than 1, blocks-per-cylinder is less than 1, or blocks-per-cylinder is greater than 500. The default values of 7 blocks-to-skip and 400 blocks-per-cylinder are used. These values were set previously when the `mkfs` (make file system) command was used to make the file system.

5.5 Using DOS-FSS and Mounting Diskette-Based File Systems

If you are using DOS-FSS and mounting a DOS file system diskette, you should know how DOS-FSS applies the UNIX System concept of permissions to a DOS file system.

Under DOS-FSS, the user who mounts a file system (usually the system administrator) is considered to be the owner of the `root` of the mounted file system. The initial permissions are 777, i.e., all users have read, write, and execute privileges. If the file system is mounted read-only, then the initial permissions are 555, i.e., all users have read and execute privileges. Once the file system is mounted, the owner can change the permissions to restrict other users' access to it.

When an attempt is made to access an individual file or directory, the permissions for that file or directory are copied from the current permissions of the `root` of the mounted directory. For example, if the permissions of the `root` of the mounted directory are set to

777 and a user accesses a file in that directory for the first time, the file's permissions will also be set to 777. When a file system is unmounted, all permissions are effectively lost and must be reinstated the next time the file system is mounted.

To mount a DOS file system on a diskette, change to, or log in as the superuser and type the command line:

```
    mount -f DOS /dev/dsk/f0q15dt /mnt
```

The file system is then mounted.

Note also that only one link is permitted to a DOS file or directory, and that any changes to the permissions or ownerships of files will be lost when the file system is unmounted.

5.6 File System Types

The INTERACTIVE UNIX Operating System is shipped with support for a variety of different file system types. Some are supplied with the base operating system while others are included with optional subsets and extensions. Each file system type has a unique structure. The UNIX System provides a consistent interface to the various file systems so that applications will be able to access files on a wide variety of local and remote storage devices.

Different file system types include network file systems such as RFS and NFS*, the DOS file system type, and two standard UNIX System Laboratories, Inc. (USL) UNIX System file system types. File system types often vary in the size of their smallest logical unit. A 1K file system type makes a logical block size of 1K. On a typical device with a physical sector size of 512 bytes, this means that one logical block is equivalent to two physical blocks. With a 2K file system type, each disk allocation or access would take place on four physical blocks.

5.6.1 Very Fast File System (VF)

The Very Fast File System (VF) is designed for high-speed sequential reading and writing of large files, such as optical image files of a megabyte or more in size.

A VF file system can be very large, up to 2 terabytes (2 to the 41st power bytes), though individual files are subject to the standard 2 gigabyte UNIX System limit. The logical block size of the VF file system is 1K bytes.

When the VF file system is used as intended, its transfer rate to and from the disk approaches the maximum transfer rate of the drive itself. The programming interfaces of the S5 and VF file systems are identical, and thus when appropriate the VF file system can transparently replace the S5 file system. The on-disk structure of the VF file system is not, however, S5-compatible, which means it cannot be used when interoperability with S5 file systems on other UNIX System platforms is required.

The improved performance of the VF file system is achieved through four techniques:

- Files are allocated in large extents. An extent is a set of 1K blocks that are contiguous on the disk.
- Large data transfers are performed directly to and from the user program address space, rather than through the buffer cache.
- Small and partial-block transfers are buffered and cached to permit read-ahead and to avoid redundant disk operations.
- The data storage area on the disk is separated into directory and data zones to reduce seek times.

Data cannot be transferred directly to or from the user address space if the first block of a read or write does not start at a block boundary and/or the last block of a read or write does not end on a block boundary. Intermediate blocks are transferred directly. The highest transfer rates are achieved when the programmer follows these rules:

- Read and write in large units.
- Read and write in block multiples on block boundaries.
- Align buffers on page boundaries.

VF file systems are created by the *mkvffs*(1M) command. Other file system administration and maintenance commands, *fsck*(1M), *fsstat*(1M), *fstyp*(1M), and *mount*(1M), recognize and correctly process VF file systems. *fsdb*(1M) cannot be used on a VF file system.

When a file in the VF file system is first written, an initial primary extent is allocated. After all the blocks in the primary extent have been written, additional secondary extents are allocated as needed. To minimize file discontinuities, a new secondary extent is allocated contiguous to the preceding extent whenever possible.

Each file has primary and secondary extent size attributes (expressed in 1K logical blocks) that determine how many blocks the operating system will attempt to allocate when a new extent is needed. If a free extent of the specified size is not available, a smaller extent will be allocated. When a VF file system is created using *mkvffs*(1M), it is given default primary and secondary extent size attributes. A file inherits these default extent size attributes when it is created. The extent sizes of a file can be changed with an *fcntl*(2) system call command as described in *vf*(4) in the *INTERACTIVE SDS Guide and Programmer's Reference Manual*, which allows a program that knows the size of the file it will write to specify a primary extent size that can contain the entire file.

Each file in a VF file system is represented by an inode structure that contains information about the file. A VF inode has entries for up to 12 extent descriptors. When a newly allocated extent is contiguous to the preceding extent, it is simply added to the last extent descriptor, minimizing the number of extent descriptors used. Thus the limit of 12 extent descriptors per file will only be a problem if the file system is badly fragmented.

5.6.2 High Sierra File System (HS)

The High Sierra file system (HS) is currently the standard format for organizing data on Compact Disk Read-Only Memory (CD-ROM) disks. ISO 9660 is an almost identical international standard. The HS file system support in the INTERACTIVE UNIX System allows a user to mount a CD-ROM disk that is in High Sierra or ISO 9660 format as an extension of the UNIX System file system.

mount(1M) recognizes and correctly mounts HS file systems if HS support is configured into the kernel. Because CD-ROM disks cannot be written, the HS file system is mounted read-only. Attempts to create, modify, or delete files or directories will fail, and the *mkfs*(1M) and *fsck*(1M) commands fail when applied to a CD-ROM. *fsdb*(1M) cannot be used on an HS file system.

The High Sierra and ISO 9660 standards specify that alphabetic characters in file and directory names must be uppercase. Lowercase characters in file names presented to the HS support system are converted to uppercase, allowing users to enter file names in lowercase.

5.6.3 The System V 1 Kilobyte File System (S51K)

The default file system type is the System V 1 Kilobyte file system (S51K). The INTERACTIVE UNIX Operating System will automatically create file systems of this type if you use either the `sysadm addhd`, `mkfdfs`, or `mkfs` facilities. The normal installation will create file systems of this type for you automatically.

The structure of this file system follows the standards for UNIX System V on personal computers. All system utilities and applications for the INTERACTIVE UNIX System are compatible with this file system type. INTERACTIVE has added enhancements to the UNIX System kernel to improve the performance of the S51K file system type. This high performance Fast File System is structurally the same as the unenhanced S51K file system. However, the UNIX System kernel manages this file system in a more intelligent fashion while remaining compatible with USL file systems.

5.6.4 The 2 Kilobyte File System Utility Package (S52K)

The optional 2 Kilobyte File System Utility Package (S52K) from USL accesses the disk in fixed, 4-block units. If it is installed and enabled using the `kconfig` program, it will:

- Decrease the available disk space
- Increase the memory requirements for the UNIX System kernel

There will be no other effects unless S52K file systems are actually created manually using the `mkfs2K` command (see *mkfs*(1M), *fsck*(1M), *fsdb*(1M), and *mount*(1M)). Since the standard versions of certain other commands and utilities are not compatible with the S52K file system type, alternate versions of these are supplied. These typically have names ending in `2K`. Not all user applications are compatible with the S52K. When using the 2K file system type:

- Performance is several times slower than with the Fast File System.
- Even the smallest files require 2048 bytes of storage.
- Mounting and unmounting times are reduced.

If you are using media with a physical block size of 2K, you must use the S52K file system type. This file system is provided with the

INTERACTIVE UNIX System mainly for consistency with USL releases.

5.6.5 The XENIX File System Type

The XENIX* file system type (XX) is used by the XENIX operating system. No provision is made in the INTERACTIVE UNIX System for creating these file systems. The INTERACTIVE UNIX Operating System does not support the mounting of any XENIX disk partitions that already exist on the fixed disk. Only diskette-based XENIX file systems can be mounted. The `xfsck` program is also provided for checking and repairing these file systems.

5.6.6 DOS File Systems

DOS file systems can be created on diskettes using the `dossette` command. Alternatively, the `mkdosfs` command can be used to create DOS file systems on both removable and fixed media. (However, you must use native DOS to create any DOS fixed disk partitions.) Both the `dosformat` command in `dossette` and the `FORMAT` utility in the optional VP/ix* Environment extension can perform a low-level diskette format and then create the DOS file system in a single operation. The `mkdosfs` command requires that the diskette be preformatted by the `format` command.

5.6.7 512

Several utilities are provided with the suffix `512`. These utilities are present for historical reasons and serve no function. It is not possible to mount diskettes created with these utilities.

5.6.8 Optical and Removable Media

Drives that support writable optical media can be prepared using the normal procedure and `sysadm addhd`. You may use `addhd` even if your writable media is contained in a removable cartridge. In some cases cartridges may need preparation or formatting using special utilities, although the INTERACTIVE UNIX System formatting procedure will correctly prepare most 512 and 1024 byte-per-sector optical cartridges. If the medium has 2K sectors, you must use the S52K file system type. If you are using a WORM drive, the fast file systems conversion from bit map to free list may use up an unacceptable number of blocks, in which case the S52K file system may be preferable.

CD-ROM and other unwritable storage devices employ their own individual file system types. The most popular file system type for

CD-ROM devices is the High Sierra file system type. Since writing on them is impossible, `mkfs` and `fsck` utilities are not supplied. These devices should always be mounted read-only. To prevent the UNIX System from becoming confused, it is important to unmount CD-ROM cartridges before removing them, just as you would any other UNIX System file system.

5.6.9 Type Names

The names of file system types that are available in your system are listed, one per file, in the directory `/etc/conf/sfsys.d`. Those files containing the file system type followed by a `Y` are enabled, and those type names followed by an `N` are disabled. The `kconfig Facilities` option under `Configure` on the bar menu will enable or disable the support for different file system types, respectively. When using the `-f` (type) flag of the `mount` command, use the file system type name exactly as it appears in these files. If the file contains a type name in uppercase letters, be sure to specify it in uppercase.

6. BACKING UP AND RESTORING FILES

Once you have started to use your system, you will need to back up your files. It is important to perform backups on a regular basis because backup copies of your files are your only protection in the event of a file system crash.

There are two kinds of backups. An *archival* (complete) backup is used to back up all the files and directories on a particular file system. An *incremental* backup is used to back up only those files and directories that have changed since the last complete backup.

You should plan a backup strategy for your system that includes a regular complete backup, supplemented by more frequent incremental backups. You may also transfer a file, a set of files, or the contents of a directory on to some other medium for storage.

Use the `sysadm File` menu to transfer data from your fixed disk to another storage medium, usually diskettes or tape.

This section explains how to back up file systems and individual files or directories on to some other medium. It also explains how to restore files you have backed up.

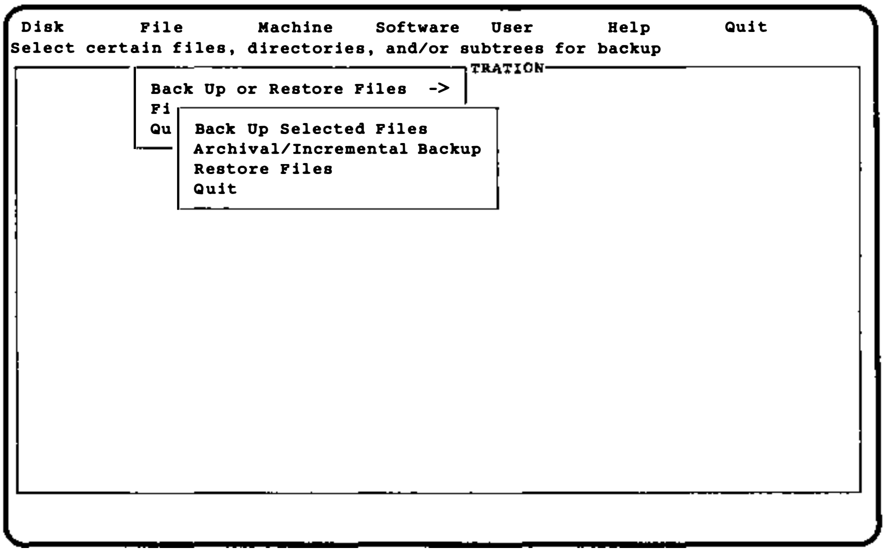
6.1 Before You Begin

The first time you perform a backup, it should be an archival one. When you do a complete backup of a file system, *all* the files on a specified file system are transferred to another medium. This may take some time to complete.

If you are backing up a file system using diskettes, you must have a supply of formatted diskettes available. Depending on the size of your file system, a single backup can require many diskettes. Be sure you have enough formatted diskettes available to complete the backup procedure. The `sysadm Disk` menu has an option for formatting diskettes. Format all the diskettes you will need before you run the `backup` procedure.

6.2 Backing Up File Systems

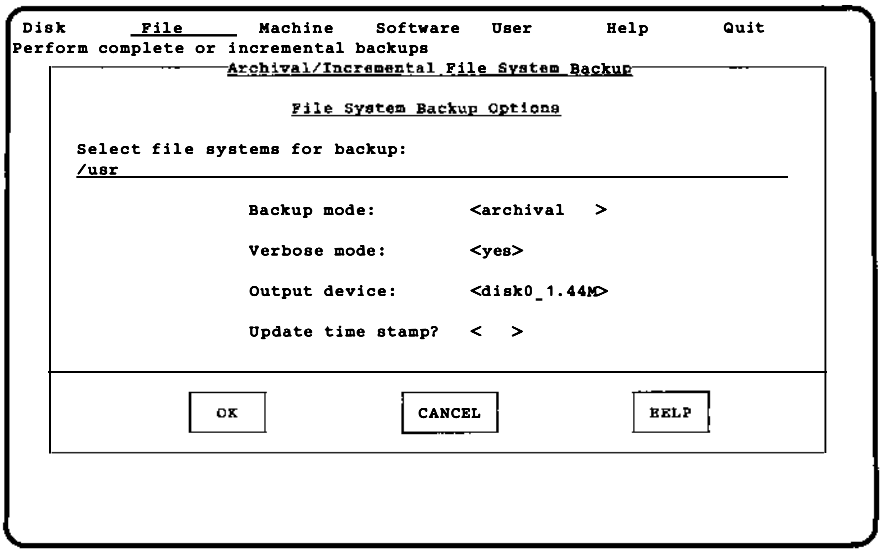
1. Access the `sysadm File` menu and select `Back Up or Restore Files.` Your screen will look similar to this:



```
Disk        File        Machine     Software    User        Help        Quit
Select certain files, directories, and/or subtrees for backup
                                                      TRATION
              Back Up or Restore Files  ->
              Fi
              Qu   Back Up Selected Files
                   Archival/Incremental Backup
                   Restore Files
                   Quit
```

If you want to back up only a few files, select `Back Up Selected Files` and skip to the next section. To perform a complete or incremental backup of file systems, move to `Archival/Incremental Backup` and press **ENTER** to select it.

2. Your screen will look similar to this:



```
 Disk      File      Machine    Software    User      Help      Quit
Perform complete or incremental backups
           Archival/Incremental File System Backup

                  File System Backup Options

    Select file systems for backup:
    /usr

                 Backup mode:         <archival   >

                 Verbose mode:        <yes>

                 Output device:       <disk0_1.44M>

                 Update time stamp?   <  >
```

3. Fill in the information required in the form to back up complete file systems. Press F1 on each field on-line to obtain more specific help.

   Select an `archival` backup to back up all the files and directories in the listed file systems. Select `incremental` to back up only the files and directories changed since a previous backup.

   If you want the backup procedure to list files during the process, select `yes` in the `Verbose mode:` field.

   Specify the output destination for the files and directories in the third field. Press the spacebar to pop up your choices. If you are backing up to the primary tape drive on your machine (`/dev/tape`), select tape; if you are using diskettes, select the appropriate drive and type of diskette. If you are backing up to any other kind of tape or using a command, select `other` and a new form will appear.

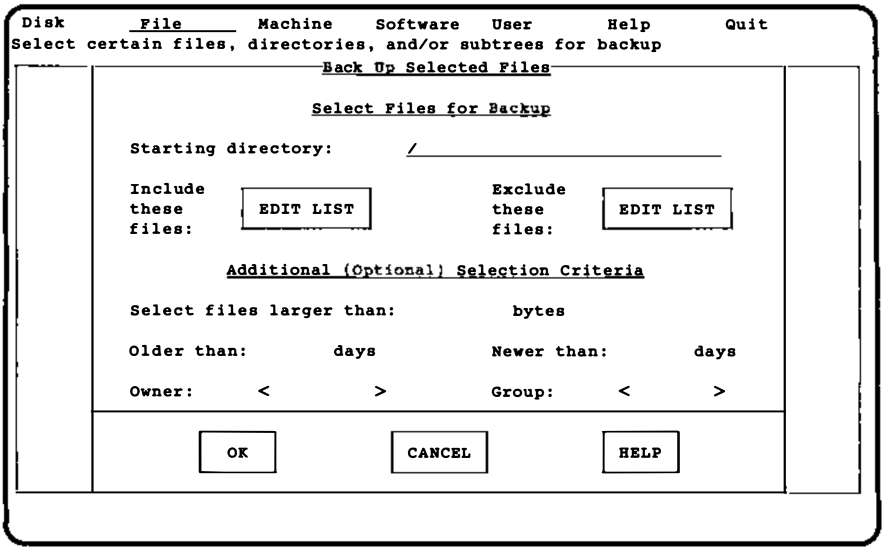
   Use the last field to update or not update the time stamp on files and directories after a backup. The time stamp must be updated when doing an archival (complete) backup.

   When you are satisfied with the information on the form, move to the OK button and press ENTER. The system copies all of the files in the named file system(s) onto the specified output device.

6.3 Backing Up Individual Files and Directories

There may be occasions when you do not want to transfer a complete file system to some other medium. You can use the `Back Up Selected Files` option of the `Back Up or Restore Files` menu to transfer a file, a group of files, or the contents of a directory to another medium.

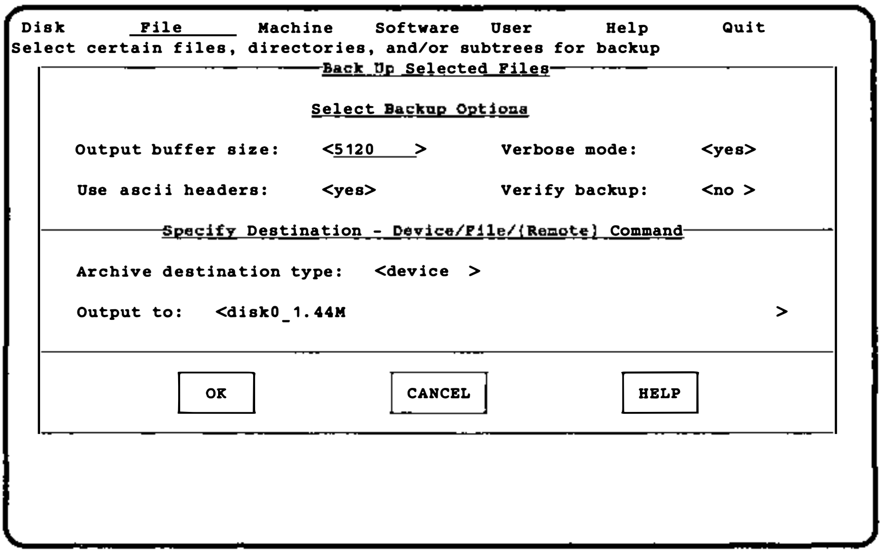
1. Select `Back Up Selected Files` from the `Back Up or Restore Files` menu under `File`. Your screen will look similar to this:



```
Disk      File      Machine    Software    User      Help      Quit
Select certain files, directories, and/or subtrees for backup
                          Back Up Selected Files

                          Select Files for Backup

    Starting directory:       /

    Include                            Exclude
    these       EDIT LIST              these       EDIT LIST
    files:                             files:

                Additional (Optional) Selection Criteria

    Select files larger than:          bytes

    Older than:          days          Newer than:          days

    Owner:       <         >           Group:       <         >

                OK            CANCEL            HELP
```

2. If you want to use a relative path name, use the first field to type in the name of the directory you want to start with. Move to the `EDIT LIST` buttons and press $\boxed{\text{ENTER}}$ if you want to call up new forms that allow you to explicitly list the files, directories, and subtrees (a subtree includes all files and directories contained within the directory) to be included and excluded in the backup. Using the appropriate fields on the form, you can back up files based on their size, age, time of last modification, owner, or group. Note that the default (`/`) is to back up all files in the diskette file system or on the fixed

disk. When you are satisfied with the information on the form, move to the OK button and press ENTER. Your screen will look similar to this:

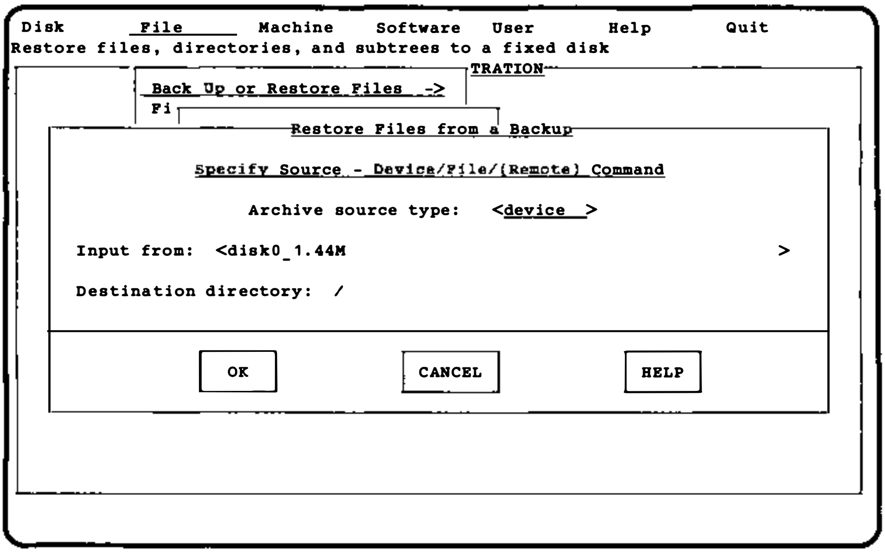


```
 Disk      File       Machine    Software   User      Help      Quit
Select certain files, directories, and/or subtrees for backup
                        Back Up Selected Files

                        Select Backup Options

   Output buffer size:     <5120    >       Verbose mode:      <yes>

   Use ascii headers:      <yes>            Verify backup:     <no >

         Specify Destination - Device/File/(Remote) Command

   Archive destination type:   <device  >

   Output to:   <disk0_1.44M                                      >

               OK                CANCEL               HELP
```

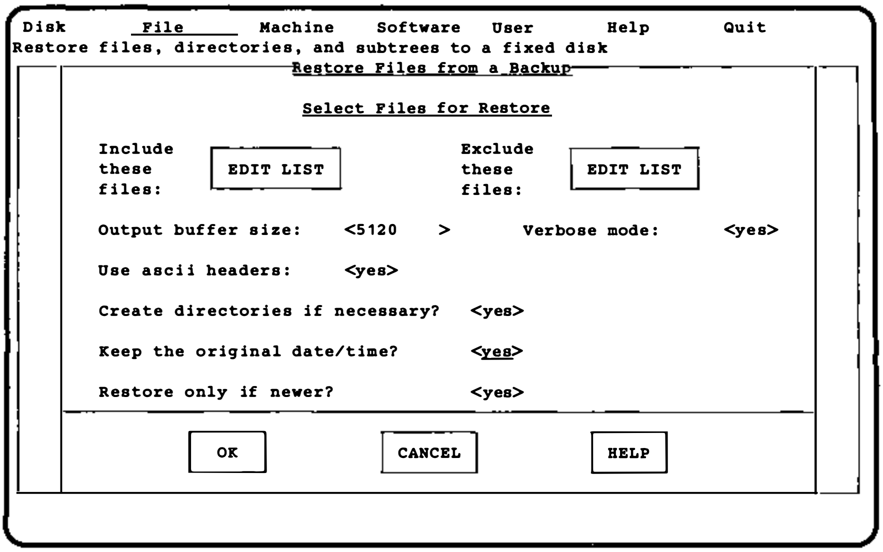
After you fill in the options you want and specify the back up destination, move to the OK button and press ENTER. The system transfers the files to the medium you have selected.

6.4 Restoring Files

To restore files you have backed up, use the `Restore Files` option on the `Back Up or Restore Files` menu under `File` on the bar menu. Your screen will look similar to this:


```
Disk       File       Machine     Software    User       Help       Quit
Restore files, directories, and subtrees to a fixed disk
                                        TRATION
               Back Up or Restore Files  ->
               Fi
                          Restore Files from a Backup

                Specify Source - Device/File/(Remote) Command

                      Archive source type:   <device  >

     Input from:  <disk0_1.44M                                        >

     Destination directory:  /

                 OK              CANCEL              HELP
```

Use this form to copy files back on to your fixed disk from a device, an archive file, or using a command, as when restoring files and directories across a network. You can optionally choose to restore the files and directories to a location different from the original one. Help for each field is available on-line. When you are satisfied with the information on the form, move to the OK button and press ENTER. Your screen will look similar to this:



```
Disk        File        Machine     Software    User        Help        Quit
Restore files, directories, and subtrees to a fixed disk
                           Restore Files from a Backup

                           Select Files for Restore

    Include                             Exclude
    these      EDIT LIST                these      EDIT LIST
    files:                              files:

    Output buffer size:   <5120    >       Verbose mode:        <yes>

    Use ascii headers:    <yes>

    Create directories if necessary?   <yes>

    Keep the original date/time?       <yes>

    Restore only if newer?             <yes>

               OK               CANCEL               HELP
```

After you fill in the options you want, move to the OK button and press ENTER. The system transfers the files.

7. USING `kconfig` TO TAILOR YOUR SYSTEM KERNEL

Your system kernel is initially configured to support a basic hardware and software configuration. The default kernel includes drivers for the following configuration:

Hardware:

- A keyboard
- A fixed disk and diskette controller
- A monochrome, color, EGA, or VGA display adapter
- Two serial communications ports (COM1, COM2)
- Up to three parallel ports
- A real time clock
- CMOS RAM
- Intel* 80387 floating point co-processor (80287 supported on COMPAQ 386*)

UNIX System V Release 3.2 Software, including:

- MS-DOS* file system service
- UNIX System V file system service
- Common interprocess communication routines, including:
  - –Interprocess communication message facility
  - –Interprocess communication semaphore facility
  - –Shared memory
- 287 or 387* floating point co-processor emulator

Your vendor may deliver a different default configuration. If your software did not come directly from INTERACTIVE, check the documentation supplied with your system to determine its default configuration.

The kernel initially installed on the INTERACTIVE UNIX Operating System supports the standard default configuration shown above. You may have already reconfigured and rebuilt your kernel if you installed certain optional subsets and extensions provided with your system. The kernel will need to be reconfigured and rebuilt to support further changes you may want to make to your system if such changes include:

- Changing the configuration in a way which involves either the addition or removal of a device driver or the reconfiguration of the High Performance Device Driver (HPDD).
- Adding or removing certain software packages.
- Changing the system memory size.
- Adding or modifying tunable system parameters.

Note that simply *installing* a driver or software package that contains a driver does *not add* that driver to your system configuration. After installing the driver you will still need to reconfigure and rebuild the kernel so that it will recognize and use the driver.

The `kconfig` program is a menu-driven interface used to configure, build, and install new kernels. The general uses of `kconfig` are discussed here. The information on specific hardware configurations is found in section 8, "HARDWARE COMPATIBILITY AND CONFIGURATION." For more specific technical information on the underlying programs and files used to configure, build, and install kernels, see *idbuild*(1M) and *inskern*(1).

The `kconfig` program is supplied with the Kernel Configuration subset, which must be installed before you can access it. To install this subset, follow the instructions in section 6.1, "Installing Optional INTERACTIVE Subsets and Extensions" in the "INTERACTIVE UNIX Operating System Installation Instructions."

The displays in this section are examples only. Your screens may differ, depending upon your software release and the drivers or facilities available on your machine.

7.1 The `kconfig` Interface

The kernel that is shipped with the INTERACTIVE UNIX Operating System is built in such a way to allow you to install many different hardware configurations. This kernel may be used successfully without alteration, but you can substantially improve your system's performance by reconfiguring, building, and installing a kernel specifically tailored to your hardware configuration. The `kconfig` program makes it easy for you to reconfigure, build, and install a kernel that reflects your system needs. You will probably want to reconfigure the HPDD, system tunable parameters, and the individual driver configuration.

To find out how to use the kconfig interface program, refer to section 2, or use the kconfig program to access the Help title on the bar menu on-line and read “Using Help” and “Using the INTERACTIVE CUI System.”

7.2 Using the kconfig Interface

You must be the superuser (root) to run the kconfig program. The kernel configuration files are located in the etc/conf directory tree under the root directory (/ by default). The root directory can be changed either by setting the environment variable $ROOT at the system prompt or by using the -r *root_directory* option on the kconfig command line. The configuration directory tree is referred to here as $ROOT/etc/conf. When running the kconfig program, the actual root directory will be substituted.

If the UIcolor environment variable has not been set, the system asks if you want to continue in color mode. Answer n to invoke the kconfig program in monochrome mode. To prevent this question from being asked each time you run kconfig, set the UIcolor environment variable for root. To do this, edit the file /.profile to add either:

```
UIcolor=TRUE
export UIcolor
```

(for color monitors) or

```
UIcolor=FALSE
export UIcolor
```

(for monochrome monitors).

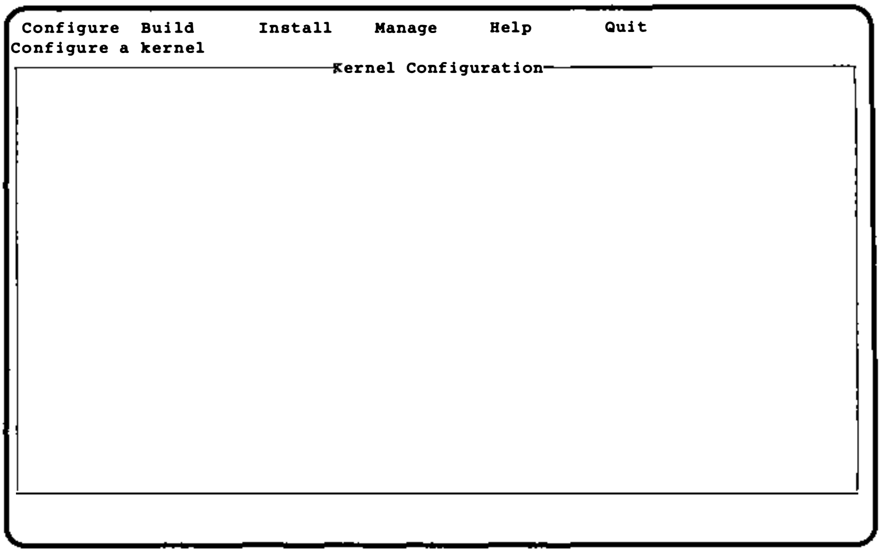
Log in as or su to root and type:

```
kconfig
```

If the root directory has not been specified, kconfig prompts:

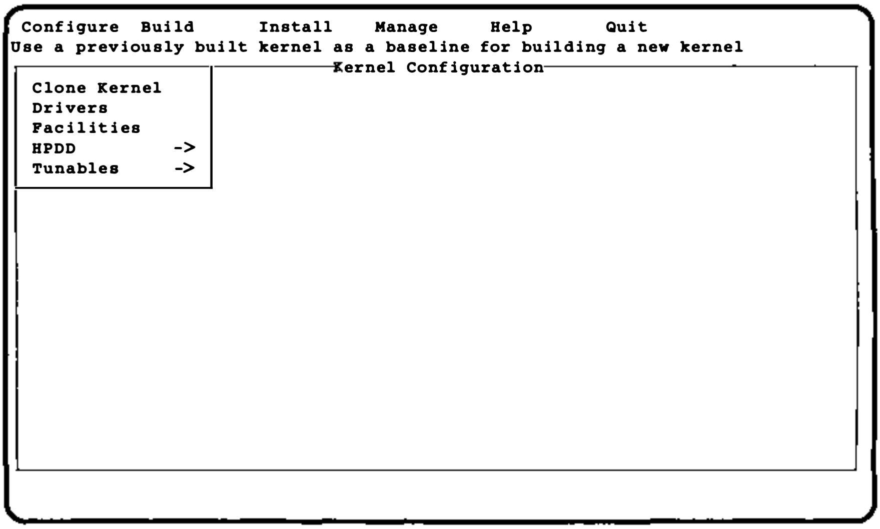
```
Root Directory (/):
```

Press ENTER to accept the default. The system displays the kconfig bar menu.



```
 Configure  Build      Install    Manage     Help       Quit
Configure a kernel
                               Kernel Configuration
```

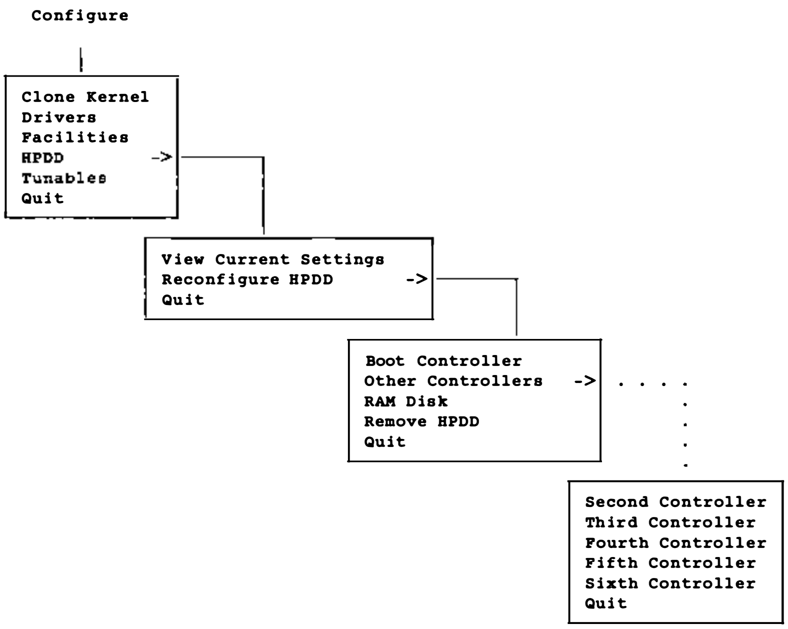
Select `Configure` from the bar menu to access the `Configure` menu.



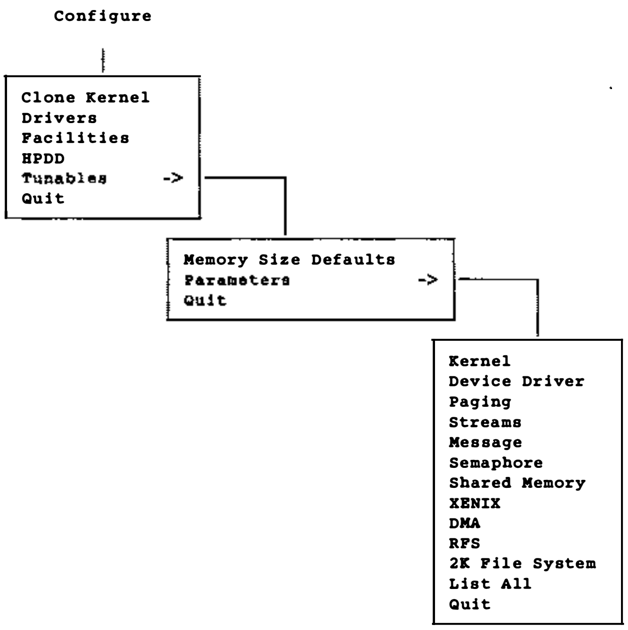
```
 Configure  Build      Install    Manage     Help       Quit
Use a previously built kernel as a baseline for building a new kernel
                               Kernel Configuration
  Clone Kernel
  Drivers
  Facilities
  HPDD         ->
  Tunables     ->
```

The options on the `Configure` menu allow you to use a previously built kernel as a baseline for building a new kernel and to configure drivers, facilities, the High Performance Device Driver, and kernel tunable parameters.

The following diagrams show the possible menu choices. To configure the High Performance Device Driver:



To configure the tunable parameters:



Each option is discussed in subsequent sections. When you exit the `Configure` menu, `kconfig` asks if you want to save the changes that have been made during the session. If you have changed your mind about the new configuration and do not want it to take effect, press ENTER while the cursor is on the NO button. If you want to save the changes, press ENTER while the cursor is on the `YES` button and the values that you configured will be saved.

7.3 Configuring the Kernel

The INTERACTIVE UNIX Operating System is delivered with a number of device drivers that support a wide variety of devices. More information about these drivers can be found in section 8, "HARDWARE COMPATIBILITY AND CONFIGURATION." Refer to the "INTERACTIVE UNIX Operating System Release Notes" for a list of the hardware devices that are supported. The kernel also contains the High Performance Device Driver, a system of device and controller drivers that together provide fast, consistent support for many disk and tape devices. Refer to section 8.2 for more information about the devices supported by the HPDD and to the

"INTERACTIVE UNIX Operating System Installation Instructions" for information on the default configuration supported by the HPDD. Many optional subsets, extensions, and facilities of the INTERACTIVE UNIX Operating System contain device drivers. (Facilities may be software extensions to the kernel that provide special capabilities, such as interprocess communication and shared memory, or they may be packages that support specific hardware, such as the SunRiver* Fiber Optic Station.) Depending on the type of driver, different mechanisms are used to configure the kernel to support it. However, you will always begin by selecting `Configure` from the `kconfig` bar menu. Each procedure is discussed in subsequent sections.

- If the device or driver is supported by the HPDD, select `Reconfigure HPDD` from the `HPDD` menu under `Configure`.
- If the driver is in the list of drivers provided with the INTERACTIVE UNIX Operating System as noted in the release notes, select `Drivers` from the `Configure` menu. This option provides a list of the drivers available for addition or removal.
- If the driver is part of a facility provided with the INTERACTIVE UNIX Operating System, it can be added or removed by selecting `Facilities` under the `Configure` menu. This option provides a list of the facilities available for addition or removal.
- To configure the system for a driver provided by a third party (i.e., not INTERACTIVE), follow the installation instructions provided by the vendor. If the driver is not part of the HPDD and the vendor used the integration scheme set up by INTERACTIVE for the INTERACTIVE UNIX System, then after installation you should be able to remove the driver by selecting `Drivers` under the `Configure` menu.

To configure the kernel, do the following:

1. Log in as or `su` to `root`, type `kconfig`, and select `Configure` from the bar menu.

   Each option provides step-by-step procedures for the indicated task. If your configuration changes result in modifications to system files, then after quitting the `Configure` menu `kconfig` asks if you want to save your changes.

2. If you have changed your mind about the new configuration and do not want it to take effect, select the `NO` button. If you want the new configuration, press ENTER to accept the default (the `OK` button). You must then build and install the new kernel for this configuration to take effect.

7.3.1 Cloning a Kernel

Select `Clone Kernel` under `Configure` on the bar menu to duplicate the configuration of an existing kernel. This procedure changes the entries in the existing kernel configuration files: `/etc/conf/sdevice.d`, `/etc/conf/cf.d/mdevice`, `/etc/conf/cf.d/mtune`, `/etc/conf/cf.d/stune`. This kernel can then be modified and used as a basis for building and installing a new kernel.

A list of the available kernels will be displayed. Select the one you want and press ENTER while the `DUPLICATE` button is highlighted. Use the `Drivers`, `Facilities`, and `Tunables` options to modify this kernel to suit your needs. If no kernels are available, a message that there are no kernels to choose from is displayed.

7.3.2 Configuring Drivers

Individual drivers may need to be configured, depending on the configuration of system hardware. Select `Drivers` under `Configure` on the bar menu to include kernel device driver modules in the set of configured modules, which may also include third-party drivers that were previously installed and configured into the system. The system displays the drivers currently available on your machine and whether each is configured in to or out of your software. To review or modify a driver, move to the one you want and press ENTER while it is highlighted. Another screen appears, describing the configuration of the driver. You can change the status of the driver (either include it or exclude it from the configuration of the next kernel you build), and you can modify individual driver parameters. The parameters that may be modified include the interrupt (IRQ) level, Direct Memory Access (DMA) channel, I/O address space, controller memory address space, and interrupt priority level. To configure a driver into the kernel configuration, move to the `Driver Status` radio button and select `on`; select `off` to configure this driver out of the kernel configuration.

☛ Note that certain drivers (the Bell Technologies Blit driver, for example) require that both driver entries be configured `on` for the driver to be fully functional. Both must also be configured out to remove them from the kernel configuration. Consult the driver documentation for any driver that you are unsure about configuring.

7.3.3 Configuring Facilities

Select `Facilities` under `Configure` on the bar menu to add or remove optional facilities that require kernel modification. Facilities may be software extensions to the kernel that provide special capabilities, such as interprocess communication and shared memory, or they may be packages that support specific hardware, such as INTERACTIVE TCP/IP.

The system displays the facilities currently available on your machine and whether each is configured in to or out of your software. To add or remove a facility, move to the one you want and press ENTER while it is highlighted. The system will ask you to confirm the addition or removal of the facility selected. Configuring a facility configures all the drivers associated with it. In some cases, INTERACTIVE TCP/IP, for example, you will also need to configure the facility's driver(s) using the `sysadm` (system administration) program.

7.3.4 Configuring the High Performance Device Driver

The devices supported by the HPDD are fixed disk controllers and SCSI tape drives (see the release notes for a list of supported devices). The HPDD also supports a RAM disk, created by reserving a portion of the computer's available memory that is treated as if it were a disk storage device. (Refer to section 7.3.4.3 for more information about RAM disks.) To support a configuration of these devices that differs from the default (documented in the "INTERACTIVE UNIX Operating System Installation Instructions"), you must reconfigure the HPDD.

Even if your hardware does not require you to reconfigure the HPDD, reconfiguring it will result in faster startup times. The kernel used to install the system and placed on the fixed disk supports many different fixed disk controllers. During system startup, the software attempts to initialize all of the supported controllers in order. If the controller is not found (because it is not part of your hardware configuration), the next one is tried until the entire list is finished. This process takes time because the system waits a certain amount of time to be sure that each controller is not responding. Reconfiguring the HPDD eliminates the time spent attempting to initialize all the supported controllers.

The `Reconfigure HPDD` menu presents options that enable you to:

- Add, remove, change, or specify the type of fixed disk controller.
- Change the interrupt vector, DMA addresses, and I/O memory addresses used by a SCSI host adapter.
- Add, remove, or change the type of tape drive connected to a SCSI host adapter.
- Add, remove, or change the size of a RAM disk.

After reconfiguring the HPDD, if you make any additional changes to your system involving these devices (other than replacing a standard AT* controller of one type with a different standard AT controller or adding or removing fixed disk drives), you will need to reconfigure the HPDD. For more complete information on the HPDD, see section 8.2.

The High Performance Device Driver has been enhanced to control up to 32 devices per controller. There are four options when configuring the HPDD: selecting a boot controller, selecting a secondary controller, selecting a RAM disk, if desired, and removing the HPDD from the kernel configuration.

☛ Note that the complete configuration of the HPDD should be performed as a single step, that is, the boot controller, additional controllers, and a RAM disk should all be configured at the same time. If you add another controller later, you must reconfigure the entire HPDD (boot controller, additional controllers, and RAM disk) again.

7.3.4.1 The Boot Controller. Select `Boot Controller` from the `Reconfigure HPDD` menu on the `HPDD` menu under `Configure`. The available controllers are displayed; move to the controller you want, and press ENTER to select it. A form then appears with the pertinent information about the selected controller. In most cases it is not necessary to change any of the fields in this form; however, the fields may be altered to match your hardware specifications. After you are satisfied with the values in the form and have pressed ENTER while on the OK button, you will be asked whether you want to use the basic or advanced configuration screen for the controller. The advanced configuration form is required *only* if there is more than one Logical Unit Number (LUN) for any particular target (SCSI ID number). Only seven available targets are presented on the basic configuration form. For the standard ESDI/RLL/MFM-type controller, select the basic form. Disks attached to these controllers correspond to target 0 for the first disk and target 1 for the second disk. Select the appropriate device for the target number on the form you selected and save your changes. The devices supported by the HPDD are magnetic disk, SCSI (Small Computer Systems Interface) tape devices, CD-ROM devices, and WORM (Write Once, Read Many) drives.

Target numbers and Logical Unit Numbers are adapted from the standards used by the Small Computer Systems Interface to describe the physical layout of the devices connected to the SCSI Host Bus Adapter. In this layout, each adapter has 0 through 7 target numbers, one of which is reserved for the adapter itself. Each of the remaining 7 target numbers can have up to eight devices attached to it. Each device is assigned a LUN number on the target. Devices are accessed by the controller through their target and LUN numbers.

7.3.4.2 Secondary Controllers. This procedure is essentially the same as configuring a boot controller. The same forms appear, in the same order. Up to five secondary controllers may be installed and configured. The only limitations are hardware considerations such as IRQ and address space conflicts.

7.3.4.3 Configuring a RAM Disk. To configure a RAM disk, select `RAM Disk` from the `Reconfigure HPDD` menu on the `HPDD` menu under `Configure`. Press the spacebar to pop up a list of sizes for the RAM disk. Move to the one you want, and press ENTER to select it. If the size you want is not on the list, select `other` and a new form appears. You can type in any size from 256 to 3840 4K pages.

7.3.4.4 Removing the HPDD From the Kernel Configuration. This option is available for third-party vendors who require that the HPDD not be configured into the system.

☛ This option should not be used under any other circumstances.

Select `Remove HPDD` from the `Reconfigure HPDD` menu on the `HPDD` menu under `Configure` and follow the instructions.

7.3.5 Configuring Tunable System Parameters

You can adjust default system parameters to improve performance. For example, you may have installed a number of applications on your system and run out of a particular resource (such as stream queues) when running them concurrently. You can select `Parameters` from the `Tunables` menu under `Configure` to increase the parameter being exceeded; however, you should be familiar with adjusting tunable parameters before using this option.

The system tunable parameters are used to set various table sizes and system thresholds to handle the expected system load. They can have a profound effect on system performance. The initial tunable parameter values delivered for your system are acceptable for most configurations and applications. If an application has special performance needs, you may have to experiment with different combinations of parameter values to find an optimal set, or a new tunable parameter may need to be defined for add-on drivers. The intended use of your computer and your observations on how well it is performing should be used as a guide in determining the need to adjust tunable parameters. To modify kernel parameters, the INTERACTIVE UNIX Operating System kernel will have to be reconfigured, built, and installed.

There are two `kconfig` options that are concerned with tunable parameters. If you have added more memory, refer to section 7.3.5.2, "Memory Size Defaults," to change a group of parameters to preset values based on the memory size. To add a new tunable parameter or to modify the value of an existing one, see section 7.3.5.1, "Parameters." After parameters or memory size have been modified, the kernel must be rebuilt and installed and the system rebooted for the new values to take effect.

*7.3.5.1 **Parameters.*** To alter the values of a kernel tunable parameter, select the `Parameters` menu from the `Tunables` menu, under `Configure`. If you are unsure of the classification of the tunable parameter you want to change, select `Miscellaneous`, which displays *all* the available tunable parameters. Move to the tunable parameter you want to alter. If you want to see a description of the parameter, press **F1**. If you want to change the current setting and default values, **TAB** to the field you want to change and type in the new value.

The tunable parameters are contained in the `mtune` and `stune` files in `$ROOT/etc/conf/cf.d`, the kernel configuration directory. Each parameter is assigned a default value, a minimum value, and a maximum value in the `mtune` file. To override the default value of a tunable parameter, the parameter is added to the `stune` file with a value that falls between the minimum and maximum values in `mtune`. These files can be examined to determine the tunable parameter settings for your computer. See *mtune*(4) and *stune*(4) in the *INTERACTIVE UNIX System User's/System Administrator's Reference Manual* for more information about these files.

Although it is not recommended that a parameter be set outside the `mtune` limits, if it is determined that a parameter must be set higher than permitted in the `mtune` file, you can edit the limits directly. Extreme care must be taken when modifying `mtune` that other values are not modified or deleted.

You should also be aware that the INTERACTIVE UNIX System kernel forces some parameters to be within preset limits. For example, the parameter `NOFILES` (number of open files per user process) is forced to fall within the 20/100 limit regardless of how the values in `mtune` and `stune` are adjusted. You should never modify an `mtune` value unless you have a full understanding of how the parameter is used in the INTERACTIVE UNIX Operating System.

7.3.5.2 Memory Size Defaults. The standard kernel supplied with the INTERACTIVE UNIX Operating System is optimized for a system with 4 MB of RAM. The INTERACTIVE UNIX System will use all of the available memory that is detected when the system is initially booted. However, if you have more memory than the standard configuration, you can substantially increase system performance by using the set of default parameters that have been optimized for the amount of memory closest to the amount on your system. This will dedicate more memory to system buffers and other kernel structures as well as increase certain process-related parameters.

You should configure the kernel for the largest memory size option that does not exceed the total amount available on your system. The INTERACTIVE UNIX Operating System will operate unreliably if the system does not have as much memory as the kernel expects. If you later increase or decrease the amount of memory on your system so that it falls into a different size category, you should reconfigure the kernel.

If the memory size chosen is an increase over the size for which the system was previously configured, and if an existing value for a parameter is larger than the value for that parameter in the defaults for the chosen memory size, the parameter retains its present higher value. Thus, if you have increased a parameter prior to using this option, the effects of your increase will not be reversed. However, if the memory size chosen is a decrease in size, all of the default values are used.

Select `Memory Size Defaults` from the `Tunables` menu under `Configure` and follow the instructions on-line.

7.3.5.3 Special Case Needs. At some time or other you will probably need to tune certain parameters for particular circumstances. One common need is to create very large files. If you are the superuser, you can modify the `ulimit` for the particular shell process that you are running as superuser. You can also modify the system `ULIMIT` parameter for all users. This parameter and other commonly encountered limits are summarized in the following table:

Improvement	*Parameters*
Improve system performance* when additional memory installed.	NBUF**, NHBUF**
Other performance-related system parameters.	NAUTOUP**, MAXSLICE, BDFLUSHR, AGEINTERVAL (also see paging parameters)
Increase system limits when additional memory installed (support more users, reduce chances of system problems at times of heavy load, etc.).	NCALL, NINODE**, NS5INODE**, NFILE**, NPROC**, NREGION**, NCLIST** (also see message, semaphore, and shared memory parameters)
Users need to create bigger files.	ULIMIT
Each user needs to open more files.	NOFILES
Each user needs to run more processes.	MAXUP**
Other system limits that may be encountered.	SHLBMAX**, FLCKREC, SPTMAP, NUMXT, NUMSXT, PRFMAX (also see STREAMS and RFS parameters)
Miscellaneous.	PUTBUFSZ, DO387CR3

* Note that increasing the size of the buffer cache will increase the chances that data frequently read will be found in memory rather than on the disk. Depending on your system usage, an increase in the chance of reusing a data block may not yield an overall system performance improvement. In some system usage scenarios, it can provide a significant performance improvement.

** This parameter is automatically adjusted when using the `Memory Size Defaults` option of `Tunables` under `Configure` on the `kconfig` bar menu. Refer to section 7.3.5.2.

7.3.5.4 Exceeding System Limits. There are cases when the INTERACTIVE UNIX System kernel will print a message on the console that system limits are being exceeded. Some of the messages are only to advise you of the problem. Others precede a system "panic," in which case additional diagnostic messages are printed, and the system hangs until you reboot it. If you encounter any of the messages in the next table, refer to the appropriate tunable parameter for additional information.

Kernel Messages and Associated Tunable Parameter	
Kernel Console Message	*Parameter*
`iget - inode table overflow`	NINODE
`Timeout table overflow`	NCALL
`File table overflow`	NFILE
`mfree map overflow` *n**	SPTMAP
`Region table overflow`	NREGION
`Configured value of NOFILES (`*n*`)`* `is less than minimum (greater than maximum)`	NOFILES
`swapdel - too few free pages`	MINASMEM
`stropen: out of streams`	NSTREAM
`stropen: out of queues`	NQUEUE

* The value *n* indicates the actual value encountered by the kernel.

7.3.5.5 Parameter Descriptions. The following sections provide a breakdown of system tunable parameters defined in `$ROOT/etc/conf/cf.d/mtune`. The parameter categories are as follows:

- Kernel Parameters
- Device Driver Parameters
- Paging Parameters
- Streams Parameters
- Message Parameters
- Semaphore Parameters
- Shared Memory Parameters
- XENIX Tunable Parameters
- Direct Memory Access (DMA) Parameters
- Remote File Sharing (RFS) Parameters
- S52K (2K File System) Parameters

Note that the Streams Parameters determine configuration of the STREAMS Facilities optional subset, the RFS Parameters determine configuration of the INTERACTIVE Network Connection Facilities optional extension, and the S52K (2K File System) parameters control configuration of the 2 Kilobyte File System Utility Package optional subset. If these packages are not installed, adjusting the parameter values will have no effect upon your system's configuration.

7.3.5.6 General Kernel Parameters. The following parameters are associated with the kernel.

`NBUF`	The `NBUF` parameter specifies how many 1K system buffers to allocate. The INTERACTIVE UNIX System buffers form a data cache - a pool of memory in which recently accessed disk data is stored. Data stored in the cache is found faster when requested than data that has to be accessed on the disk, so "cache hits" reduce the number of disk accesses and thus may improve overall performance. Cache hit rate increases with the number of buffers. The entries usually range from 500 to 2000. Each buffer contains 1076 bytes. For optimal performance, 1K hash buffers (`NHBUF`) should be increased along with system buffers (`NBUF`).
`NCALL`	The `NCALL` parameter specifies how many call-out table entries to allocate. Each entry represents a function to be invoked at a later time by the clock handler portion of the kernel. This value must be greater than 2 and usually ranges from 30 to 70, with a default value of 50. Each entry contains 16 bytes. Software drivers may use call entries to check hardware device status. When the call-out table overflows, the system crashes and displays the following on the console: `PANIC: Timeout table overflow`
`NINODE`	The `NINODE` parameter specifies how many inode table entries to allocate. Each table entry represents an in-core inode that is associated with an active file, such as a current directory,

	an open file, or a mount point. Changing this variable modifies the file control structure. The number of entries used depends on the number of opened files. Entries usually range from 400 to 800. The value for NINODE is directly related to NFILE, as NINODE must be equal to or greater than NFILE. NINODE should always be equal to NS5INODE and to NDOSINODE. If NINODE is greater than NS5INODE, the system will be unusable. When the inode table overflows, the following is sent to the console: `WARNING: i-node table overflow`
NS5INODE	NS5INODE should always be equal to the value to NINODE and to NDOSINODE.
NDOSINODE	The NDOSINODE parameter specifies the number of inode table entries to allocate for a mounted DOS file system. This value and the values of NINODE and NS5INODE should always be equal.
NFILE	The NFILE parameter specifies how many open file table entries to allocate. Each entry represents an open file. Entries usually range from 400 to 800. Each entry contains 12 bytes. NFILE relates to the NINODE entry and should be less than or equal to NINODE. The NFILE control structure operates in the same manner as the NINODE structure. When the file table overflows, the following warning message is displayed on the system console: `NOTICE: file table overflow` As a reminder, this parameter does not affect the number of open files per process (refer to the NOFILES parameter).
NMOUNT	The NMOUNT parameter specifies how many mount table entries to allocate. Each entry represents a mounted file system. The root (/) file system is always the first entry. When the table is full, the mount command fails with an

	error message. Since the mount table is searched linearly, this value should be as low as possible.
NPROC	The NPROC parameter specifies how many process table entries to allocate. Each table entry represents an active process. The swapper is always the first entry, and /etc/init is always the second entry. The number of entries is determined by the number of terminal lines available and the number of processes spawned by each user. The average number of processes per user ranges from two to five (refer to MAXUP, default value 30). When the table is full, the fork system call fails and displays an error message and you will be unable to execute additional commands. The NPROC entry usually ranges from 100 to 200.
NREGION	The NREGION parameter specifies how many region table entries to allocate. Each NREGION entry contains 36 bytes. Most processes have three regions: text, data, and stack. Additional regions are needed for each shared memory segment and shared library (text and data) attached. However, the region table entry for the text of a "shared text" program is shared by all processes executing that program. Each shared memory segment attached to one or more processes uses another region table entry. A good starting value for this parameter is about 3.5 times NPROC. If the system runs out of region table entries, the following message is displayed on the console: `Region table overflow`
NCLIST	The NCLIST parameter specifies how many character list buffers to allocate. Each buffer contains up to 64 bytes. The buffers are dynamically linked to form input and output queues for the terminal lines and other slow-speed devices. The average number of buffers needed per terminal ranges from 5 to 10. Each entry (buffer space plus header) contains 72 bytes. When all

	`clist` buffers are full, input and output characters dealing with terminals are lost, although echoing continues.
`MAXUP`	The `MAXUP` parameter specifies how many concurrent processes an ordinary user is allowed to run, usually from 30 to 50. This value should not exceed the value of `NPROC`. `NPROC` should be at least 10 percent more than `MAXUP`. This value is a per user identification number, not per terminal. For example, if 12 people are logged in on the same user identification, the default limit would be reached very quickly.
`NOFILES`	The `NOFILES` parameter specifies the maximum number of open files per process. The default is 60. Unless an application package recommends that `NOFILES` be changed, the default setting of 60 should be used.
	The Bourne shell uses three file table entries: standard input, standard output, and standard error (0, 1, and 2 are usually reserved for `stdin`, `stdout`, and `stderr`, respectively). Thus, the usable value of `NOFILES` is three less than shown as the number of other open files available per process. If a process requires up to three more than this number, the standard files must be closed. *This should not be done.*
	If the configured value of `NOFILES` is greater than the maximum (100) or less than the minimum (20), the configured value is set to 100 in the first case and 20 in the second, and a `NOTICE` message is sent to the console.
`NHBUF`	The `NHBUF` parameter specifies how many "hash buckets" to allocate for 1K buffers. These are used to search for a buffer by a device number and block number rather than making a linear search through the entire list of buffers. *This value must be a power of 2.* Each entry contains 12 bytes. The `NHBUF` value must be chosen so that the value `NBUF` divided by `NHBUF` is approximately equal to 4.

NPBUF	The NPBUF parameter specifies how many physical I/O buffers to allocate. One I/O buffer is needed for each physical read or write currently active. Each entry contains 52 bytes. The default value is 20.
NAUTOUP	The NAUTOUP parameter specifies the number of seconds between automatic file system updates. A system buffer is written to the fixed disk when it has been memory-resident for the interval specified by the NAUTOUP parameter. Specifying a smaller limit increases system reliability by writing the buffers to disk more frequently and decreases system performance. Specifying a larger limit increases system performance at the expense of reliability. This parameter, along with BDFLUSHR, controls the behavior of the bdflush daemon process.
BDFLUSHR	The BDFLUSHR parameter specifies the rate in seconds for checking whether the file system buffers need to be written to the disk. The default is 1 second. This parameter, along with NAUTOUP, controls the behavior of the bdflush daemon process.
MAXPMEM	The MAXPMEM parameter specifies the maximum amount of physical memory (in pages). The default value of 0 specifies that all available physical memory be used.
SHLBMAX	The SHLBMAX parameter specifies the maximum number of shared libraries that can be attached to a process at one time.
FLCKREC	The FLCKREC parameter specifies the number of records (areas in a file) that can be locked by the system. The default value is 100. Each entry contains 28 bytes.
PUTBUFSZ	The PUTBUFSZ parameter specifies the size of a circular buffer, putbuf, that is used to contain a copy of the last PUTBUFSZ characters written to the console by the operating system.

The contents of `putbuf` can be viewed using the `crash` command.

`MAXSLICE` The `MAXSLICE` parameter specifies in clock ticks the maximum time slice for user processes. After a process executes for its allocated time slice, that process is suspended. The operating system then dispatches the highest priority process and allocates to it `MAXSLICE` clock ticks. `MAXSLICE` is usually the number of clock ticks in 1 second.

`ULIMIT` The `ULIMIT` parameter specifies, in 512-byte blocks, the size of the largest file that an ordinary user may create or write. The default value is 4096; that is, the largest file an ordinary user may write is 2 megabytes. The superuser may write a file as large as the file system can hold. The `ULIMIT` parameter does not apply to reads; any user may read a file of any size.

`UAREAUS` and `UAREARW`

The `UAREAUS` and `UAREARW` parameters control the ability of a user process to read and write its user area in the kernel. These parameters can be set to zero or one (the default), with the following effects:

UAREAUS	UAREARW	*User Area Accessibility*
0	0	cannot be read or written
0	1	cannot be read or written
1	0	entire area can be read, but not written
1	1	entire area can be read, first page can be written

As far as is known, no combination of these parameters compromises fundamental system security. The first page of the user area contains only the kernel stack and floating point state save area. However, some administrators may prefer to configure their systems with the user area inaccessible to the user. Note the following constraints: 1) If the system has a 386 CPU and does not have a 387 floating point coprocessor, the user area must be readable and writable. `UAREARW` and `UAREAUS` must be set to one (default). 2) If the system has a 387 floating point coprocessor and the older B1 stepping (version) of the 386 CPU, the U-area must be readable. Set `UAREAUS` to one and `UAREARW` to zero. 3) If the system has a 486 CPU, or a DX, SX, or later stepping of the 386 CPU and a 387 floating point coprocessor, the user area need not be accessible. Set `UAREAUS` and `UAREARW` to zero.

`SPTMAP`	The `SPTMAP` parameter determines the size of the map entry array used for managing kernel virtual address space. *You should not modify this parameter.*
`PIOMAP`	The `PIOMAP` parameter determines the size of the map entry array used by the kernel programmed I/O (PIO) breakup routine. This routine allows device drivers to do programmed I/O of large data blocks at interrupt level by breaking the data blocks into smaller data units. *You should not modify this parameter.*
`PIOMAXSZ`	The `PIOMAXSZ` parameter determines the maximum number of pages to use at one time for programmed I/O. *You should not modify this parameter.*
`DO387CR3`	The `DO387CR3` parameter controls the setting of high-order bits of Control Register 3 (CR3) when an 80387 math co-processor is installed.

7.3.5.7 Device Driver Parameters. The following parameters control various data structure sizes and other limits in base system device drivers.

`NUMXT`	The `NUMXT` parameter determines the number of layers a subdevice can configure to support bitmapped display devices such as the BLIT or the AT&T 5620 terminal.

NUMSXT	The NUMSXT parameter determines the number of shell layers a subdevice can configure.
NCPYRIGHT	The NCPYRIGHT parameter defines the size of a kernel data structure used to print console initialization messages. *You should not modify this parameter.*
NKDVTTY	The NKDVTTY parameter determines the number of virtual terminals (ttys) supported by the console keyboard driver. *You should not modify this parameter.*
PRFMAX	The PRFMAX parameter determines the maximum number of text symbols that the kernel profiler (/dev/prf) will be able to properly process.
TTHOG	The TTHOG parameter determines the terminal input buffer threshold. It sets the number of characters that can be entered into a tty line buffer and remain unprocessed, before the data is discarded.
RAMDSIZE	The RAMDSIZE parameter determines the maximum size of a RAM disk. The default maximum is 3840 4K pages or 15 MB.

7.3.5.8 Paging Parameters. A paging daemon, vhand, is responsible for freeing up memory as the need arises. It uses a "least recently used" algorithm to approximate process working sets, and it writes out those pages that have not been modified during some period of time to the disk. The page size is 4096 bytes. When memory is exceptionally tight, the working sets of entire processes may be swapped out.

The following tunable parameters determine how often vhand and bmapflush run and under what conditions. The default values should be adequate for most applications.

AGEINTERVAL	The AGEINTERVAL parameter specifies the number of clock ticks a process runs before its pages are aged.
VHNDFRAC	The VHNDFRAC parameter determines the initial value for the system variable VHANDL. VHANDL is set to the maximum user-available

	memory divided by `VHNDFRAC`, or the value of `GPGSHI`, whichever is larger. The value of `VHANDL` determines when the paging daemon `vhand` runs. The amount of available free memory is compared with the value of `VHANDL`. If free memory is less than `VHANDL`, the paging daemon `vhand` is awakened. The default is 16. Decreasing the value makes the daemon more active. The value must be between 0 and 25 percent of available memory.
`GPGSLO`	The `GPGSLO` parameter specifies the low-water mark (in pages) of free memory before `vhand` will start to steal pages from processes. The default is 25. Increase the value to make the daemon more active; decrease the value to make the daemon less active (this value must be an integer greater than or equal to 0 and less than `GPGSHI`).
`GPGSHI`	The `GPGSHI` parameter specifies the high water mark (in pages) of free memory before `vhand` will stop stealing pages from processes. The default is 40. Increase the value to make the daemon more active; decrease the value to make the daemon less active. The value must be an integer greater than `GPGSLO`, and less than 25 percent of the number of pages of available memory.
`GPGSMSK`	The `GPGSMSK` parameter is a mask used by the paging daemon. The default is 1056. *This value should not be changed.*
`MAXSC`	The `MAXSC` parameter specifies the maximum number of pages that will be swapped out in a single operation. The default value is 1.
`MAXFC`	The `MAXFC` parameter specifies the maximum number of pages that will be added to the free list in a single operation. The default value is 1.
`MAXUMEM`	The `MAXUMEM` parameter specifies the maximum size of a user's virtual address space in

	pages. This value cannot be greater than 8192. The default is 4096.
`MINARMEM`	The `MINARMEM` parameter specifies the minimum number of memory pages reserved for the text and data segments of user processes.
`MINASMEM`	The `MINASMEM` parameter is a threshold value that specifies the number of memory and swap pages reserved for system purposes (unavailable for the text and data segments of user processes).
`MINHIDUSTK`	The `MINHIDUSTK` parameter specifies the minimum data relocation value that allows the user stack and data to share a page table. *This value should not be changed.*
`MINUSTKGAP`	The `MINUSTKGAP` parameter specifies the minimum data relocation value that allows the user stack and data to share a page table. *This value should not be changed.*

7.3.5.9 Streams Parameters. The following tunable parameters are associated with Streams processing. The values have no effect on the system unless the STREAMS Facilities optional subset is installed.

`NQUEUE`	The `NQUEUE` parameter determines the number of Streams queues to be configured. Queues are always allocated in pairs, so this number should be even. A minimal Stream contains four queues (two for the Stream head, two for the driver). Each module pushed on a Stream requires an additional two queues. A typical configuration value is `4*NSTREAM`.
`NSTREAM`	The `NSTREAM` parameter determines the number of "Stream-head" (stdata) structures to be configured. One is needed for each Stream opened, including both Streams currently open from user processes and Streams linked under multiplexers. The recommended configuration value is highly application-dependent, but a value ranging from 32 to 40 usually suffices for running a single transport provider with moderate traffic.

`NSTRPUSH`	The `NSTRPUSH` parameter determines the maximum number of modules that may be pushed onto a Stream. This is used to prevent an errant user process from consuming all of the available queues on a single Stream. By default this value is 9, but in practice, existing applications have pushed at most four modules on a Stream.
`NSTREVENT`	The `NSTREVENT` parameter determines the initial number of Stream event cells to be configured. Stream event cells are used for recording process-specific information in the `poll` system call, in the implementation of the STREAMS `I_SETSIG` ioctl, and in the kernel `bufcall()` mechanism. A minimum value to configure is the expected number of processes simultaneously using the `poll` system call times the expected number of Streams being polled per process, plus the expected number of processes expected to use Streams concurrently. The default is 256. Note that this number is not necessarily a hard upper limit on the number of event cells that will be available on the system (see `MAXSEPGCNT`).
`MAXSEPGCNT`	The `MAXSEPGCNT` parameter determines the number of additional pages of memory that can be dynamically allocated for event cells. If this value is 0, only the allocation defined by `NSTREVENT` is available for use. If the value is not 0 and if the kernel runs out of event cells, under some circumstances it will attempt to allocate an extra page of memory from which new event cells can be created. `MAXSEPGCNT` places a limit on the number of pages that can be allocated for this purpose. Note that once a page has been allocated for event cells, it cannot be recovered later for use elsewhere. It is recommended that the `NSTREVENT` value be set to accommodate most load conditions and that `MAXSEPGCNT` be set to 1 to handle exceptional load cases.

`NMUXLINK`	The `NMUXLINK` parameter determines the maximum number of multiplexer links to be configured. One link structure is required for each active multiplexer link (STREAMS `I_LINK ioctl`). This number is application-dependent; the default allocation guarantees availability of links.
`STRMSGSZ`	The `STRMSGSZ` parameter determines the maximum allowable size of the data portion of any Streams message. This should usually be set just large enough to accommodate the maximum packet size restrictions of the configured Streams modules. If it is larger than necessary, a single `write` or `putmsg` can consume an inordinate number of message blocks. The recommended value of 4096 is sufficient for existing applications.
`STRCTLSZ`	The `STRCTLSZ` parameter determines the maximum allowable size of the control portion of any Streams message. The control portion of a `putmsg` message is not subject to the constraints of the minimum/maximum packet size, so the value entered here is the only way of providing a limit for the control part of a message. The recommended value of 1024 is more than sufficient for existing applications.
`NBLK`*n*	`NBLK4` through `NBLK4096` control the number of Streams data blocks and buffers to be allocated for each size class. Message block headers are also allocated based on these numbers; the number of message blocks is 1.25 times the total of all data block allocations. This provides a message block for each data block, plus some extras for duplicating messages (kernel functions `dupb()` and `dupmsg()`). The optimal configuration depends on both the amount of primary memory available and the intended application.
`STRLOFRAC`	The `STRLOFRAC` parameter determines the percentage of data blocks of a given class at

	which low priority block allocation requests are automatically failed. For example, if STRLOFRAC is 40 and there are 48 256-byte blocks, a low priority allocation request will fail when more than 19 256-byte blocks are already allocated. The parameter is used to help prevent deadlock situations by starving out low priority activity. A recommended value of 80 works well for current applications. STRLOFRAC must always be in the range greater than or equal to 0 and less than or equal to STRMEDFRAC.
STRMEDFRAC	The STRMEDFRAC parameter determines the percentage cutoff at which medium priority block allocations are failed (see STRLOFRAC discussion). The recommended value of 90 works well for current applications. STRMEDFRAC must always be in the range greater than or equal to STRLOFRAC and less than or equal to 100. ☛ Note that there is no cutoff fraction for high priority allocation requests; it is effectively 100.
NLOG	The NLOG parameter determines the number of minor devices to be configured for the log driver; the active minor devices will be 0 through (NLOG−1). The recommended value of 3 services an error logger (`strerr`) and a trace command (`strace`), with one left over for miscellaneous usage. If only an error logger and a tracer are to be supported, this number can be set to 2. If there are several daemons for an application that may be submitting log messages, this number can be increased to accommodate the extra users.
NUMSP	The NUMSP parameter determines the number of Streams pipe devices (`/dev/sp`) supported by the system. *You should not modify this parameter.*
NUMTIM	The NUMTIM parameter determines the maximum number of Streams modules that can be pushed by the Transport Library Interface (TLI).

	This value controls the number of data structures used to hold pushed Streams modules configuration data. *You should not modify this parameter.*
`NUMTRW`	The `NUMTRW` parameter determines the number of Transport Library Interface (TLI) read/write data structures to allocate in kernel data space. *You should not modify this parameter.*

7.3.5.10 Message Parameters. The following tunable parameters are associated with interprocess communication messages:

`MSGMAP`	The `MSGMAP` parameter specifies the size of the control map used to manage message segments. The default value is 100. Each entry contains 8 bytes.
`MSGMAX`	The `MSGMAX` parameter specifies the maximum size of a message. The default value is 2048. Although the maximum possible size the kernel can process is 64 kilobytes −1, the `mtune` limit is 8192.
`MSGMNB`	The `MSGMNB` parameter specifies the maximum length of a message queue. The default value is 4096.
`MSGMNI`	The `MSGMNI` parameter specifies the maximum number of message queues system wide (ID structure). The default value is 50.
`MSGSSZ`	The `MSGSSZ` parameter specifies the size, in bytes, of a message segment. Messages consist of a contiguous set of message segments large enough to fit the text. The default value is 8. The value of `MSGSSZ` times the value of `MSGSEG` must be less than or equal to 131,072 bytes (128 kilobytes).
`MSGTQL`	The `MSGTQL` parameter specifies the number of message headers in the system and therefore the number of outstanding messages. The default value is 40. Each entry contains 12 bytes.
`MSGSEG`	The `MSGSEG` parameter specifies the number of message segments in the system. The default

value is 1024. The value of MSGSSZ times the value of MSGSEG must be less than or equal to 131,072 bytes (128 kilobytes).

7.3.5.11 Semaphore Parameters. The following tunable parameters are associated with interprocess communication semaphores:

SEMMAP	The SEMMAP parameter specifies the size of the control map used to manage semaphore sets. The default value is 10. Each entry contains 8 bytes.
SEMMNI	The SEMMNI parameter specifies the number of semaphore identifiers in the kernel. This is the number of unique semaphore sets that can be active at any given time. The default value is 10. Each entry contains 32 bytes.
SEMMNS	The SEMMNS parameter specifies the number of semaphores in the system. The default value is 60. Each entry contains 8 bytes.
SEMMNU	The SEMMNU parameter specifies the number of undo structures in the system. The default value is 30. The size is equal to 8 times (SEMUME + 2) bytes. (Refer also to SEMUME.)
SEMMSL	The SEMMSL parameter specifies the maximum number of semaphores per semaphore identifier. The default value is 25.
SEMOPM	The SEMOPM parameter specifies the maximum number of semaphore operations that can be executed per semop system call. The default value is 10. Each entry contains 8 bytes.
SEMUME	The SEMUME parameter specifies the maximum number of undo entries per undo structure. The default value is 10. The size is equal to 8 times SEMMNU bytes. (Refer also to SEMMNU.)
SEMVMX	The SEMVMX parameter specifies the maximum value a semaphore can have. The default value is 32767, which is the maximum value for this parameter.

`SEMAEM`	The `SEMAEM` parameter specifies the adjustment on exit for maximum value, alias `semadj`. This value is used when a semaphore value becomes greater than or equal to the absolute value of `semop`, unless the program has set its own value. The default value of 16384 is the maximum value for this parameter.

7.3.5.12 Shared Memory Parameters. The following tunable parameters are associated with interprocess communication shared memory:

`SHMMAX`	The `SHMMAX` parameter specifies the maximum shared memory segment size. The default value is 1048576.
`SHMMIN`	The `SHMMIN` parameter specifies the minimum shared memory segment size. The default value is 1.
`SHMMNI`	The `SHMMNI` parameter specifies the maximum number of shared memory identifiers system wide. The default value is 100. Each entry contains 52 bytes.
`SHMSEG`	The `SHMSEG` parameter specifies the number of attached shared memory segments per process. The default value is 6. The maximum value is 15.
`SHMALL`	The `SHMALL` parameter specifies the maximum number of in-use shared memory text segments. The default value is 512.

7.3.5.13 XENIX Tunable Parameters. The following describes the XENIX tunable parameters:

`DSTFLAG`	The `DSTFLAG` parameter specifies the `dstflag` described for the XENIX `ftime`(S) system call.
`NSCRN`	The `NSCRN` parameter specifies the maximum number of virtual terminals that can be used by VT and console drivers.
`NEMAP`	The `NEMAP` parameter specifies the maximum number of I/O translation mappings.

`TIMEZONE`	The `TIMEZONE` parameter specifies the `timezone` setting referred to in the XENIX `ftime`(S) system call. Note that the timezone value is a system default timezone and not the value of the `TZ` environment variable.
`XSEMMAX`	The `XSEMMAX` parameter specifies the maximum number of XENIX special semaphores allowed system wide. The minimum value for `XSEMMAX` is 20; the maximum value (and default) is 60.
`XSDSEGS`	The `XSDSEGS` parameter specifies the maximum number of XENIX special shared data segments allowed system wide. The minimum value for `XSDSEGS` is 1; the maximum value (and default) is 25.
`XSDSLOTS`	The maximum number of XENIX special shared data segment attachments system wide is `XSDSEGS` X `XSDSLOTS`. The minimum value for `XSDSLOTS` is 1; the maximum value (and default) is 3.

7.3.5.14 Direct Memory Access (DMA) Parameters. The following parameters are associated with Direct Memory Access.

`DMAEXCL`	The `DMAEXCL` parameter specifies whether simultaneous DMA requests are allowed. Some computers have DMA chips that malfunction when more than one allocated channel is used simultaneously. For all installations on these computers, `DMAEXCL` is set to 1 by default. On computers that do not have this problem, set `DMAEXCL` to 0 to allow simultaneous DMA on multiple channels.

7.3.5.15 Remote File Sharing (RFS) Parameters. There are several parameters you can tune to best suit the way you use Remote File Sharing (RFS). RFS parameters control the amount of system resources you devote to RFS service. Each network transport provider may also have some tunable parameters that may affect performance characteristics of that particular network. See the network documentation for your network for more details.

All parameters have set default values that should work well for an average system; however, if the values are too small, you may not be providing enough resources to properly handle your RFS load. Requests for mounts, advertises, or even a file could fail if either of those values reach the maximum number allowed for your machine. If these parameters are too large, you could be allocating more system resources than you need to use.

Note that these parameters have no effect on your system unless the INTERACTIVE Network Connection Facilities optional extension is installed.

NRCVD	The NRCVD parameter specifies the maximum number of receive descriptors. Your system creates one receive descriptor for each file or directory being referenced by remote users and one for each process on your machine awaiting response to a remote request. If you limit the number of receive descriptors, you limit the number of local files and directories that can be accessed at one time by remote users. Exceeding the limit results in error messages for remote user commands.
NSNDD	The NSNDD parameter specifies the maximum number of send descriptors. For each remote resource (file or directory) your users reference, your system creates a send descriptor. A send descriptor is also allocated for each server process and each message waiting in the receive queue. You can change this value to limit how many remote files and directories your machine can access at a time. In effect, this limits the RFS activities your users can perform. Exceeding the limit results in error messages for user commands.
NSRMOUNT	The NSRMOUNT parameter specifies the number of server mount table entries. Each time a remote machine mounts one of your resources, an entry is added to your server mount table. This number limits the total number of your resources that can be mounted at a time by remote machines.

`NADVERTISE`	The `NADVERTISE` parameter specifies the number of entries in the advertise table. An entry is placed in your advertise table for each resource you advertise. This parameter sets the maximum number of resources you can advertise.
`MAXGDP`	The `MAXGDP` parameter specifies the number of virtual circuits allowed. There are up to two connections (virtual circuits) set up on the network between you and each machine with which you are currently sharing resources. There is one for each computer whose resources you mount and one for each computer that mounts your resources. A virtual circuit is created when a computer first mounts a resource from another, and it is taken down when the last resource is unmounted. This parameter limits the number of RFS virtual circuits your computer can have open on the network at a time. It limits how many remote computers you can share resources with at a time. Note that a given network may have a limited number of circuits on any one computer, so this parameter influences the maximum percentage of those that can be used for RFS.
`MINSERVE`	The `MINSERVE` parameter specifies the minimum number of server processes. Your system uses server processes to handle remote requests for your resources. This parameter determines the number of active server processes on your computer.
`MAXSERVE`	The `MAXSERVE` parameter defines the maximum number of server processes. When there are more remote requests for your resources than can be handled by the minimum servers, your computer can temporarily create more. This parameter sets the maximum total server processes your system can have (`MINSERVE` plus the number it can dynamically create).
`NRDUSER`	The `NRDUSER` parameter specifies the number of receive descriptor `user` entries to allocate.

Each entry represents a client machine's use of one of your files or directories. While there is one receive descriptor allocated for each file or directory being accessed remotely (NRCVD), there can be multiple receive descriptor `user` entries for each client using the file or directory (NRDUSER). These entries are used during recovery when the network or a client goes down. This value should be about 1.5 times the value of NRCVD.

RFHEAP

The RFHEAP parameter specifies the size in bytes of an area of memory set aside for RFS information. It contains the following information:

- The user and group ID mapping tables and the domain name of each machine currently sharing a resource(s) with your machine.
- A list of machine names supplied as a client list when you advertise resources.

  The appropriate size for RFHEAP is affected by:

  – UID/GID tables (size and number).

    There will always be two global tables, one UID and one GID. Also, any machine with a `host` entry in the `uid.rules` or `gid.rules` files will have a table corresponding to each of these entries while it is connected to this machine. Machines that do not have separate entries in one of these files do not take any extra space.

    To estimate the size on an individual table, type `idload -n`. There will be one 4-byte table entry per line of output from `idload`, plus up to 24 bytes of overhead per table.

- Adv client lists (size and number).

  Each advertise may have a list of authorized clients attached to it. This list is stored in this area, with its size unchanged, until the resource is unadvertised.

- Currently connected resources.

  Each connection will use a maximum of 64 bytes to store the name of the connected resource. This memory is allocated dynamically, so some additional space is required to account for possible fragmentation as space is allocated and de-allocated. Since the total size is likely to be relatively small, 1 to 4 kilobytes, it is best to allow too much rather than too little space.

NLOCAL

The NLOCAL parameter sets the minimum number of local access buffers, available from the common buffer pool, reserved for local access. RFS client caching shares the common buffer pool with the local accesses (usually disk or tape). Therefore, this value protects local data from adverse effects of competition with RFS buffer use.

When this threshold is turned off (set to 0), it defaults to the recommended value of one third of the entire buffer pool (NBUF). A nonzero value of NLOCAL overrides this default. Note that if RFS is not running or has had no recent activity, the entire buffer pool will be available to local access.

`NREMOTE`	The `NREMOTE` parameter sets the minimum number of local buffers, available from the common buffer pool, reserved for remote resource read data. When this threshold is turned off (set to 0), it defaults to the recommended value of one third of the entire buffer pool (`NBUF`). A nonzero value of `NREMOTE` overrides this default. The sum of `NREMOTE` and `NLOCAL` must not be greater than `NBUF`. If this condition is detected, a console warning message is printed, and the default value (one third of `NBUF`) is used for both `NREMOTE` and `NLOCAL`.
`RCACHETIME`	The `RCACHETIME` parameter can be used in two ways: to turn off caching for your entire machine or to define the number of seconds that network caching is turned off when a file is modified. To turn off caching for your entire machine, set this parameter to −1. The second use of `RCACHETIME` requires some explanation. When a write to a server file occurs, the server machine sends invalidation messages to all client machines that have the file open. The client machines remove data affected by the write from their caches. Caching of that file's data is not resumed until the writing processes close the file or until the seconds in this parameter have elapsed. The assumption is that write traffic occurs in bursts and that the first write may be closely followed by other writes. Turning off caching avoids the overhead of sending invalidation messages for subsequent writes.
`RFS_VHIGH`	The `RFS_VHIGH` parameter determines the highest RFS version number with which your machine will communicate.
`RFS_VLOW`	The `RFS_VLOW` parameter determines the lowest RFS version number with which your machine will communicate.

In addition to the above, the `NHBUF` parameter has implications for RFS. The value of `NHBUF` is used to specify how many "hash buckets" to allocate for remote data in the buffer pool, as well as for local data. The hash buckets are used to search for a buffer given a remote server machine ID and file ID, rather than a linear search through the entire list of buffers. (See section 7.3.5.6, "General Kernel Parameters," for further discussions of `NHBUF`.)

7.3.5.16 S52K (2K File System) Parameters. Note that these parameters have no effect on your system unless the 2 Kilobyte File System Utility Package optional subset is installed.

`S52KNBUF`	The `S52KNBUF` parameter specifies how many 2K system buffers to allocate. This parameter performs the same function for 2K file systems that `NBUF` performs for 1K file systems. The entries usually range from 100 to 400. Each buffer contains 2100 bytes. 2K hash buffers (`S52KNHBUF`) should be increased along with `S52KNBUF` for optimal performance. If you configure 2K buffers in your system, you should reduce the number of 1K buffers (`NBUF`) to keep available memory at an acceptable level.
`S52KNHBUF`	The `S52KNHBUF` parameter specifies how many hash buckets to allocate for 2K buffers. These are used to search for a buffer by device number and block number rather than making a linear search through the entire list of buffers. *This value must be a power of 2.* Each entry contains 12 bytes. The `S52KNHBUF` value must be chosen so that the value of `S52KNBUF` divided by `S52KNHBUF` is approximately equal to 4.

7.4 Building a New Kernel

Select `Build` from the bar menu to review the configuration of the most recently configured kernel or to build that kernel.

After you have made changes to the current kernel configuration, you need to build a new kernel, install it in the appropriate directory, shut down the system, and reboot it.

Log in as or `su` to `root` and type `kconfig`. Select `Build` from the bar menu. The system kernel is automatically built, using the information you have already provided. If the kernel builds successfully, the following message is displayed:

```
Finished building unix system
```

After a successful kernel build, the system prompts you to type in a description of the kernel. Type something that will help you differentiate this kernel from other kernels you build. You may want to mention the drivers and facilities configured in to this kernel. If you do not type in a description, the time and date of the build is used.

The system uses the modified configuration files to build a new kernel in a subdirectory under `$ROOT/etc/conf/kconfig.d`. The subdirectory has the name of the new kernel with `.d` appended to the end. In this example, the new kernel is `unix.1` and is located in `$ROOT/etc/conf/kconfig.d/unix.1.d`. The new kernel is linked to the `root` directory (/) if possible; otherwise it is copied there. When it is installed, it is linked from the `root` directory to the `/unix` file, overwriting the current copy of the kernel contained in `/unix`. Thus, `/unix` always reflects the latest installed version of the kernel. When the system is rebooted, the new kernel is in effect.

If the kernel fails to build, error messages are displayed. You can scroll through them using the up and down arrow keys. Correct the errors and build the kernel again.

7.5 Installing a Previously Built Kernel

The `kconfig` program keeps a copy of every kernel you build in a subdirectory under `$ROOT/etc/conf/kconfig.d`. You may have configured and built a new kernel but not yet installed it, or, at times, you may want to install one of the kernels that you have used in the past. You might want to install a previous version of the kernel if, for example:

- A new device driver has been installed, but the new kernel proves to be unreliable.
- You remove a facility or driver and want to return to a kernel that is not configured for that facility or driver.

After a kernel is built, it must be installed in order to have the configuration values take effect. To install a previously built kernel, type `kconfig` to access the bar menu and select `Install`. The system displays all the kernels contained in the

`$ROOT/etc/conf/kconfig.d` directory. Move to the one you want and press ENTER to select it. If the description line is insufficient to identify the kernel you want, you may review the precise configuration of any available kernel by highlighting the one you want to review in the list and then selecting the `REVIEW` button. The exact configuration of the selected kernel is then displayed. The description of this kernel may also be altered at this time.

If you are running `kconfig` from the console, the system will ask if you want to run `shutdown`. You will have the option to type in the time in seconds before the system will shut down and reboot with the newly installed kernel.

7.6 Managing Kernels

Select `Manage` from the bar menu to remove kernels from the system or to generate a report on a specific kernel. When a kernel is generated, it is placed in a file called `$ROOT/etc/conf/kconfig.d/unix.`*n*`.d`, along with all the configuration files that were used to create it. Use the `Remove a Kernel` option to remove an old kernel, the configuration files, and this directory, freeing file system space. The configuration of the kernel may be reviewed before removal.

If the changes to the system include changes in the hardware configuration, such as the addition of a new device, then after configuring, building, and installing the kernel and shutting down the system, you should:

- Power down the system.
- Physically install the hardware device (unless it is already present as a component of your basic system hardware) and set any jumpers or switches that are necessary.
- Use the manufacturer's *setup* program to configure your system, if necessary.
- Reboot the system.

7.6.1 Removing a Previously Built Kernel

Kernels use a large amount of space on your system. If you build several kernels, you may want to periodically delete those you do not anticipate using in the future, in order to free up space on your system. Select `Remove a Kernel` under `Manage` on the bar menu to remove previously built kernels.

7.6.2 Reporting On a Previously Built Kernel

A report generated from the configuration files of a previously built kernel may be produced by selecting `Report on Kernels` under `Manage` on the bar menu. The report contains a description of the kernel, the status of the drivers and facilities, HPDD information, and all the tunable system parameter information. This report may be directed to the screen, a file, or directly to a printer (`/bin/lp`). To direct it to a file, select `File` in the form and a new form appears, asking for the full path name for the generated report. After completing the form, select the `GENERATE` button to obtain the report.

7.7 What to Do If the System Does Not Boot

There is a remote possibility that a new kernel you have installed will not boot properly. For example, this can happen if a system parameter or some combination of parameters you have modified has built a kernel that will not initialize properly. In such an event, do the following:

1. Reboot the system.
2. When the message

   ```
   Booting the UNIX System...
   ```

   is displayed, quickly press the spacebar and type `/OLD.unix` when prompted for the name of the kernel.
3. After the system is up, move the file named `/OLD.unix` to `/unix` and then move `/etc/conf/cf.d/OLD/*` to `/etc/conf/cf.d`. See *inskern*(1) for further information about which files were moved or copied during the installation of the new kernel.

8. HARDWARE COMPATIBILITY AND CONFIGURATION

8.1 Introduction

The hardware mentioned in this section is not meant to be an exhaustive list of the systems that are believed to be compatible with the INTERACTIVE UNIX Operating System or an endorsement of these systems. It is supplied for purposes of information only. No guarantees are expressed or implied. The definitive list of supported hardware can be found in the INTERACTIVE UNIX Hardware Compatibility Guide, which can be obtained by calling the INTERACTIVE Teleservices Department at (800) 346-7111. Additional information about driver configuration files can be found in the manual entries *mdevice*(4) and *sdevice*(4).

8.2 High Performance Device Driver

8.2.1 Overview

The High Performance Device Driver (HPDD) is a system of device and controller drivers that together provide fast, consistent support of many disk and tape devices. Underneath the HPDD "surface" are tape and disk device drivers that support operations for each of these device types. These drivers communicate via an HPDD-supplied interface to the controllers that actually drive their devices.

Tape devices on Small Computer System Interface (SCSI) host bus adapters are supported by the `tape` driver in the HPDD, except for the COMPAQ* SCSI tape adapter, which is *not* handled by the HPDD (refer to section 8.8). Non-SCSI tape devices are handled by other drivers, which are independent of the HPDD. (Refer to sections 7.5 and 7.6 for more information about the drivers that handle non-SCSI tape devices.) Diskettes are also handled separately. (Refer to section 8.3, "The Diskette Driver," for more information about diskette devices.) MFM, RLL, ESDI, IDE, and SCSI disks, on either AT or Micro Channel* architecture (MCA) buses, are handled by the disk driver, `dsk`, which also provides a RAM disk software driver.

Disk controllers for MFM, RLL, IDE, and ESDI drives on AT bus machines are handled by the `athd` module. On an MCA bus machine, MFM and RLL disk controllers use `mcst`, and ESDI controllers use `mcesdi`. SCSI host bus adapter modules are specific to their respective cards. Refer to section 8.2.2 to find out how the adapter module names correlate with supported adapters. RAM disks are controlled by `gramd`.

The HPDD now uses the SCSI naming convention to address cascaded peripheral systems with multiple devices. In earlier releases of the HPDD, only the first four available disks and first four available tape drives were accessible by users. This was insufficient to handle the 32 devices that can be connected to a SCSI bus running off a host bus adapter (HBA). Each HBA can connect up to seven target controllers, each of which can have eight devices (logical units). Each logical unit is assigned a number, which is known as a LUN or *l*ogical *u*nit *n*umber for short.

Under the new scheme, the HPDD can access up to 56 devices per HBA. Physical devices are addressed by the same HBA/target/LUN combination defined in the SCSI standards. The same scheme is applied to the AT attachment systems. An AT disk controller can be attached to a SCSI HBA and a direct couple disk to an embedded SCSI device with a fixed target/LUN combination. For example, the secondary disk on the primary AT fixed disk controller can be represented using the SCSI convention as HBA 0, target 1, LUN 0.

8.2.2 Compatibility

The HPDD supports the following classes of controllers:

Controller Module	*Manufacturer*	*Controller*
ami	Olivetti	SCSI controller
escx	American Megatrends, Inc.	BT-742 in extended mode
athd	Various	All ST-506, ESDI, RLL, and IDE controllers on ISA or EISA machines
	COMPAQ	IDA in compatibility mode
dpt20xx	DPT	PM 20*xx* in extended mode
ida	COMPAQ	IDA in extended mode
mcst	Various	All ST-506 controllers on MCA machines
mcesdi	Various	All ESDI controllers on MCA machines
mcis	IBM*	IBM PS/2* MCA SCSI adapter
aha1520	Adaptec*	AHA-1520, AHA-1522
aha1540	Adaptec	AHA-154*x*, AHA-164*x*, AHA-1740 in compatibility mode
	BusTek	BT-742 in compatibility mode
aha1740	Adaptec	AHA-1740 in extended mode
bt742	BusTek	BT-742 in extended mode
tmc8x0	Future Domain	Various Future Domain SCSI adapters
wdasc	Western Digital	WD-7000ASC – FASST WD-7000EX in compatibility mode
wdex	Western Digital	WD-7000EX in extended mode
ncr700	NCR	NCR 53C700 SCSI I/O Processor
ami	Olivetti	SCSI controller
escx	American Megatrends, Inc.	BT-742 in extended mode

Note that the ncr700 driver is a single-threaded driver that was developed to support the Intel L-series systems with an on-board 32-bit NCR 53C700 SCSI controller (see *ncr700*(7)). The NCR 53C700's I/O base address is at 0xcc0.

8.2.3 Hardware Configuration

The installation kernel is configured to recognize all of the Industry Standard Architecture (ISA) and Micro Channel Architecture (MCA) adapters listed in this section. The Extended Industry Standard Architecture (EISA) modules, ida, aha1740, bt742, and wdex, are not configured in to the installation kernel because these adapters can be operated in compatibility mode. In this mode, they behave like their ISA counterparts. To use their extended mode capabilities, the extended mode module must be loaded using the kconfig program. In the case of aha1740 and wdex, EISA configuration is also needed.

The Adaptec AHA-1520 and Western Digital WD-7000ASC have an I/O address conflict. The Western Digital WD-7000ASC can only boot at base address `0x350`. The standard Adaptec AHA-1520 can only boot at base address `0x340`, and the on-board registers extend into the address space of the WD-7000ASC. If the boot adapter is the standard Adaptec AHA-1520, a secondary WD-7000ASC may only exist at a base address other than `0x350`. If the boot adapter is a WD-7000ASC, you cannot use a standard Adaptec AHA-1520 as a secondary adapter. To remove the conflict, Adaptec can supply a nonstandard AHA-1520 adapter with an I/O base address of `0x140`. The driver in the installation kernel will automatically find the adapter at the new address.

The Adaptec AHA-154*x* and Western Digital WD-7000ASC adapters can only access physical memory smaller than 16 MB, which is a problem when using these adapters in EISA machines with more than 16 MB of memory. To circumvent this problem, the `aha1540` and `wdasc` drivers can restrict the use of more than 16 MB of memory. If either driver is loaded and found to have devices attached, memory greater than 16 MB is disabled. These drivers are loaded in the installation kernel and may be used by EISA adapters in their compatibility mode. Memory truncation can be avoided by replacing Adaptec AHA-154*x* and Western Digital WD-7000ASC adapters with their EISA counterparts and by running EISA adapters in their extended mode. This usually requires that the kernel be reconfigured using the `kconfig` program.

EISA Adapter	*Behaves Like*
Adaptec AHA-1740	Adaptec AHA-1540
BusTek 742	Adaptec AHA-1540
WD-7000EX	WD-7000ASC
IDA	ATHD
escx	ATHD
ami	BT-742

The default configuration is:

Installation Kernel Configuration			
Controller Module	*I/O Addresses*	*IRQ*	*DMA Channel*
`athd`	0x1f0-0x1f7,0x3f6	14	None
`tmc8x0`	None	5	None
`aha1520`	0x340	11	None
`aha1540`	0x330	11	5
`mcis`	0x3510	14	None
`mcst`	0x320	14	3
`mcesdi`	0x3510	14	5
`wdasc`	0x350	15	6
`dpt20xx`	0x230 (ISA only)	14	5 (ISA only)
`ncr700`	0xcc0	11	None (EISA only)

The HPDD can deal with as many adapters as there are slots available in the machine. An arbitrary limit of six is imposed by the `kconfig` program. There are limits on the number of adapters of a particular type that are allowed in any configuration. Adapters under the control of the `athd`, `mcst`, and `msesdi` drivers are limited to two per configuration. Other than that, the only limiting factors are hardware resources such as interrupt vectors, DMA channels, and I/O address space.

While the theoretical limit on the number of devices on a SCSI adapter is 56, 32 is the limit imposed by table sizes within the HPDD. Non-SCSI adapters are limited to two disks.

8.2.4 Configuring the High Performance Device Driver

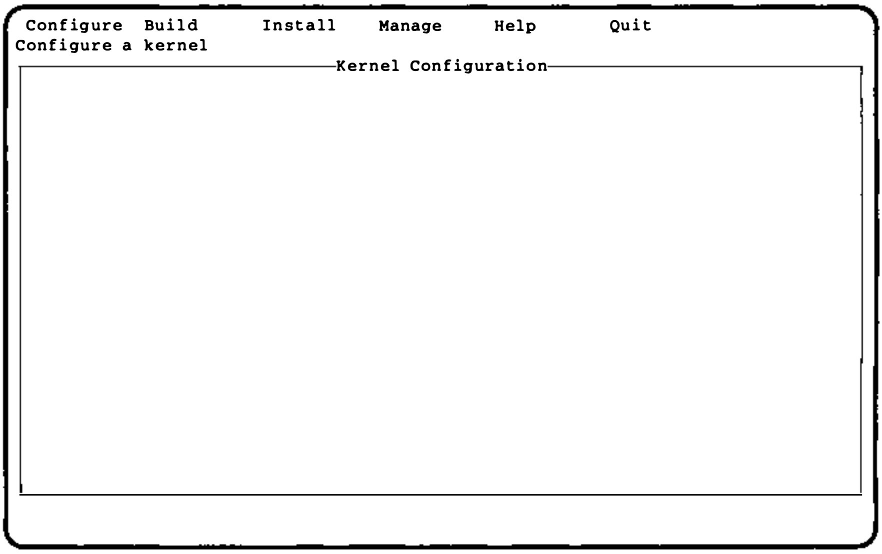
If you add or remove a device that is supported by the HPDD, you will need to reconfigure it. Regardless of adding or removing devices, it is a good idea to configure the HPDD after you install your system because it will significantly increase the speed of booting the system.

⇒ Note that you must already have installed the Kernel Configuration optional subset in order to configure the HPDD.

To configure the HPDD, read the preceding sections carefully to determine any special requirements. Log in to the system as `root` (or use `su`), and type `kconfig` to invoke the `kconfig` program. If the `root` directory has not been specified, `kconfig` prompts:

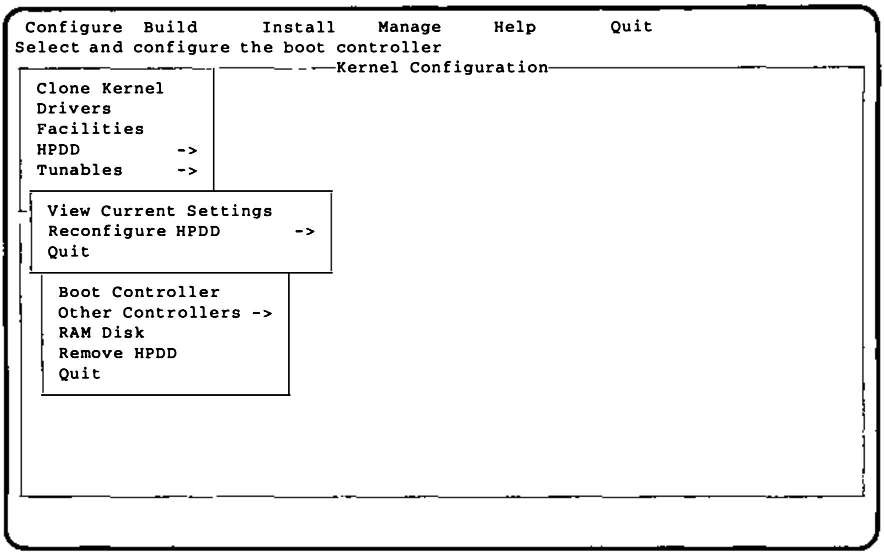
```
   Root Directory (/):
```

Press $\boxed{\text{ENTER}}$ to accept the default. The system displays the `kconfig` bar menu.



```
 Configure  Build       Install    Manage     Help       Quit
Configure a kernel
                              Kernel Configuration
```

Select `Configure` from the bar menu to access the `Configure` menu. Select `HPDD`, then `Reconfigure HPDD`, and follow the instructions on-line for the changes you want to make. Refer to section 7.3.4 for more information about reconfiguring the HPDD.



When you exit the `Configure` menu, `kconfig` asks if you want to save the changes that have been made during the session. If you have changed your mind about the new configuration and do not want it to take effect, press ENTER while the cursor is on the NO button. If you want to save the changes, press ENTER while the cursor is on the `YES` button and the values that you configured will be saved.

You will then need to build and install the new kernel as described in sections 7.4 and 7.5. If you also plan to add new drivers for unsupported devices at this time, or otherwise reconfigure the kernel, you may want to do those tasks first and then build and install the kernel after all changes are done.

8.2.5 Auto-Configuration

Some modules in the HPDD, `aha1520`, `aha1540`, `aha1740`, `bt742`, `ida`, `dpt20xx`, `escx`, `ncr700`, and `wdex` can override the values configured in `dsk/space.c` and `tape/space.c` to match values presented by the adapter itself. During initialization, these modules look at the configuration information in the EISA NonVolatile Random Access Memory (NVRAM) or, in the case of `aha1520` and `aha1540`, some on-board registers. This information is checked against the software configuration in `dsk/space.c` and `tape/space.c`. If the information matches, initialization proceeds as normal. Several passes through the software configuration are made in order to get the best possible match. During pass 1, adapters are assigned positions if the software configuration matches the hardware configuration exactly. During pass 2, adapters are assigned positions if the software I/O base address matches the hardware I/O base address. During pass 3, adapters are assigned positions if the software interrupt vector matches the hardware interrupt vector. During pass 4, unallocated adapters are assigned based on slot order.

Auto-configuration can alter the software configuration to match the hardware configuration, which can have some detrimental effects on system operation. When auto-configuration changes an adapter's I/O base address, the change is transparent because many EISA adapters have slot-dependent I/O base addresses, so simply moving an adapter to a different slot should not prevent the system from booting. An interrupt vector mismatch, however, is not transparent. The interrupt vector given to the HPDD for an adapter may already be in use by another device. In this case, the interrupt is stolen from the other device for use by the HPDD, and the other device can no longer operate. When the interrupt vectors do not match, a screen that looks similar to this will appear during initialization:

```
--------------- HPDD IDA NOTICE ---------------
Kernel's IDA interrupt vector=15
EISA NV RAM configuration IDA interrupt vector=11
These two interrupts should be the same number!
Run kconfig, reconfigure the HPDD, and rebuild the kernel
-------- AUTO CONFIGURATION TAKING OVER --------
```

This should be resolved quickly by either giving the named adapter (in this case the Intelligent Drive Array (IDA)) a new interrupt using the EISA configuration utility (or an on-board jumper for non-EISA boards) or by changing the software configuration to match the hardware configuration using the `kconfig` program, or both.

8.2.6 Warnings

The following drivers provided with the INTERACTIVE UNIX Operating System are known to conflict with the default configuration of the HPDD:

Driver	*Conflict*
`logi`	Uses interrupt 5, conflicts with `tmc8x0`
`mouse`	Uses interrupt 5, conflicts with `tmc8x0`
`wt`	Uses interrupt 5, conflicts with `tmc8x0`

If any of these drivers are to be used in the same kernel as the HPDD configured with a conflicting controller, the conflict must be resolved by rejumpering one of the boards and manually changing the corresponding driver configuration files. See *mdevice*(4) and *sdevice*(4) for details about the configuration files. Remember that if you are not actually using the conflicting HPDD controller, configuring the HPDD as described in section 8.2.5 will resolve the conflict.

Quarter inch tape drives connected to the Adaptec SCSI host adapter function only at their native densities; for example, a 150 MB drive will not read or write a 60 MB tape.

8.2.7 Device Names

8.2.7.1 Disks. There are two sets of rules for device name construction as it applies to disks. The set you should use depends on the type of pseudo device (target or LUN) to which the real device is connected. If the connection is to a target pseudo device, `dsk0t`, for example, the device names follow the forms `c`*i*`t`*j*`p`*k* and `c`*i*`t`*j*`s`*k*, where *i* is the controller number (0 in this case because `dsk0t` is being used), *j* is the target on the designated controller, and *k* is the partition number on that drive. All numbers are zero-based.

If the connection is to a LUN pseudo device, `dsk32`, for example, the device names follow the forms `c`*i*`t`*j*`l`*k*`p`*m* and `c`*i*`t`*j*`l`*k*`s`*m*, where *i* is the controller number (3 in this case because `dsk32` is being used), *j* is the target number on the controller (2 in this case because `dsk32` is being used), *k* is the LUN on the designated target on the designated controller, and *m* is the partition number (`0-f`) on that drive. All numbers are zero-based.

When a LUN pseudo device is being used, the first device on the designated target will have two names – the LUN-based name and the target-based name. This occurs because the target interface always accesses the first LUN on any target.

By default, the first two targets on the primary adapter have additional names constructed of three characters: a digit representing the target number (0 or 1), the letter p for non-remapped fdisk partitions or the letter s for remapped INTERACTIVE UNIX System partitions, and a hexadecimal digit for the partition number (0–9, a–f).

It is recommended that you use the full names for the sake of clarity.

The fdisk partitions 1–f correspond to those shown by the fdisk program. The fdisk table itself can only have four partitions, but DOS supports a sub-partitioning scheme via extended partitions. The first 11 of these are available through the suffixes p5 through pf.

The INTERACTIVE UNIX System partitions 1–f correspond to those in the UNIX System VTOC structure, maintained by the mkpart program (see *mkpart*(1M). These partitions contain file systems, swap area, alternate sectors, the INTERACTIVE UNIX System bootstrap, and other utility areas.

Partitions p0 and s0 are special cases. The p0 device provides the entire disk device; the s0 device provides the entire INTERACTIVE UNIX System partition, which corresponds to one of the devices p1 through p4, depending on how the disk was partitioned with fdisk, except that bad sectors are mapped out.

Each disk pseudo driver has its own major numbers for the block and character interfaces. Major numbers are normally assigned by the kconfig program and both numbers should be the same. The target interface to the primary adapter (dsk0t) is always major number 0.

The minor numbers associated with device names are constructed of three fields. The high order bit (0x80) is the p bit and is 1 if a p device is being referenced. The next three bits (0x70) have a different meaning, depending on the type of pseudo driver attached. If the pseudo driver is a target driver (for example, dsk4t – adapter 4, first LUN on each target), then the three bits refer to the target number. If the pseudo driver is a LUN driver (for example, dsk46 – adapter 4, all LUNs on target 6), then the three bits refer to the LUN number on the designated target. The last field (0x0f) is the partition number and corresponds to the last digit of the device name.

The RAM disk, if configured, is simply another target pseudo driver and has only one minor number, `0`.

The following table contains some examples of device names and their minor numbers. Note that the major numbers are not known for many devices since these are assigned by the `kconfig` program and thus vary from system to system.

Sample HPDD Device Names and Minor Numbers

Device Name	*Major Number*	*Minor Number*	*Description*
`0p0`	0	0x80	The whole disk on the first disk on the primary controller.
`c0t0p0`	0	0x80	Full name for `0p0`.
`0s1`	0	0x01	Usually the INTERACTIVE UNIX System `root` file system on the first disk on the primary controller.
`c0t0s0`	0	0x00	Full name for `0s0`.
`1s1`	0	0x11	The first INTERACTIVE UNIX System partition on the second disk on the primary controller.
`c0t1s1`	0	0x11	Full name for `1s1`.
`c3t5s3`	Varies	0x53	Fourth adapter, target 5, INTERACTIVE UNIX System partition 3.
`c1t2l3s4`	Varies	0x34	Second adapter, target 2, LUN 3 INTERACTIVE UNIX System partition 4.
`ram`	Varies	0x00	The RAM disk.

8.2.7.2 Special Disk Information. This version of the INTERACTIVE UNIX Operating System supports CD-ROM drives on the SCSI bus. The following CD-ROM drives are known to work:

- Sony* Model CDU-621
- NEC Model CDR-80, CDR-82
- Toshiba Model TXM-3201A1

Raw device names appear in the `/dev/rdsk` directory. Device names depend on host bus adapter number (HBA), SCSI target ID numbers (SCSI ID), and an optional logical unit number (LUN). Device names have the following syntax:

```
cd<HBA>t<SCSI ID>[l<LUN>]
```

The CD-ROM major number is handled by pseudo device major numbers.

The minor numbers associated with CD-ROM device names are constructed from three fields. The high order bit (0x80) is the *p* bit; it is always 1 since with CD-ROM the entire physical disk is always referenced. The next three bits are the device index field (0x70). The last field is the partition number field (0x0f) and is always set to zero since CD-ROM disks do not use partitions. As with other devices, minor numbers can be set differently by configuring the HPDD manually.

All known SCSI CD-ROM drives have integrated SCSI controllers that only communicate with a single CD-ROM drive. Until controllers that support multiple devices are developed, CD-ROM drives must have a unique SCSI ID for a given HBA, and the logical unit number is generally internally set to 0. This means that the only devices that should be set to CD-ROM on the HPDD SCSI configuration screen are targets 0–6, all with LUN 0.

8.2.7.3 Tapes

This version of the INTERACTIVE UNIX Operating System supports cartridge tape drives on the SCSI bus (see *tape*(7)). The following cartridge tape drives are known to work:

- Archive* Model 2060, 2125, or 2150
- Cipher 9-Track
- Wangtek Model 5099, 5125, or 5150
- Exabyte EXB-8200 and EXB-8500

As with disks, there are two sets of rules for tape device name construction. The set used depends on the type of pseudo device to which the real device is connected.

If the connection is to a target pseudo device, `tp0t`, for example, the device names are of the forms `rmt`*i*`t`*j* and `nrmt`*i*`t`*j* where *i* is the controller number (`0` in this case because `tp0t` is being used) and *j* is the target on the designated controller. All numbers are zero-based. The initial `n` indicates that the tape will not rewind when closed.

If the device is connected is to a LUN pseudo device, `tp32`, for example, the device names are of the form `rmt`*i*`t`*j*`l`*k* `nrmt`*i*`t`*j*`l`*k*, where *i* is the controller number (`3` in this case because `tp32` is being used), *j* is the target number on the controller (2 in this case because `tp32` is being used), and *k* is the LUN on the designated target on the designated controller. All numbers are zero-based.

When a LUN pseudo device is being used, the first device on the designated target has two names – the LUN-based name and the target-based name. This occurs because the target interface always accesses the first LUN on any target.

Each tape pseudo driver has its own major numbers for the character interface. Major numbers are normally assigned by the `kconfig` program. The minor numbers associated with device names are constructed from three fields. The three bits (`0x70` or the fifth, sixth, and seventh bits) have a different meaning, depending on the type of pseudo driver attached. If the pseudo driver is a target driver (`tp4t`, for example, which is adapter `4`, first LUN on each target), then the three bits refer to the target number. If the pseudo driver is a LUN driver (`tp46`, for example, which is adapter `4`, all LUNs on target `6`), then the three bits refer to the LUN number on the designated target. The second field (`0x0c`) is the tape behavior field and determines the "rewind on close" (0x80, the fourth bit) and "return immediately" (0x40, the third bit) characteristics of the unit. The last field (`0x03`) sets the density for the tape and is usually `0`.

The flag bits are:

HPDD Tape Minor Number Flag Bits

Bit Mask	*Meaning*
0x08	Rewind on close. If not set, write a filemark or forward to next filemark on close, if opened for writing or read-only, respectively.
0x04	Return immediately. Tape motion *ioctl()* calls and closings should return immediately. If not set, these calls wait for tape motion to complete before returning.
0x03	Tape density. See the next table for the meaning of this field.

The tape density field above is defined as follows:

HPDD Tape Density Values

Descriptor	*Value*	*Density Description* *9-Track BPI*	*Cartridge Tracks/Size*
Default	0x00	Default density (defined by device driver)	
Low	0x01	800 BPI	9 track (60MB)
Medium	0x02	1600 BPI	15 track (125MB)
High	0x03	6250 BPI	18 track (150MB)

The tape device names created when configuring via the `kconfig` program depend on which pseudo devices exist. In any case, one tape must be designated the default. In addition to its formal names, this tape will also be known by the names in the following table.

SCSI Tape Devices

Name	*Description*
`/dev/ntape`	No rewind after closing
`/dev/rnmt0`	No rewind after closing
`/dev/tape`	Rewind after closing
`/dev/rmt0`	Rewind after closing
`/dev/rSA/tape`	Rewind after closing

Tape devices are created using the `node.d` file with the same name as the pseudo driver to which the device is connected. If the `kconfig` program is always used to configure the kernel, it should not be necessary to modify these files. In the case of conflict with other tape devices (the Wangtek tape driver, `wt`, for example), it may be necessary to modify a pseudo driver `node.d` file or the `/etc/conf/node.d/wt` file.

For example, if you have a Wangtek (`wt`) tape drive and a SCSI tape drive (for example, target 1 on the first adapter) on your system and you want the SCSI tape drive to be the default drive, edit `/etc/conf/node.d/wt` so that it includes the following:

```
wt      rnmt0   c       0
wt      rmt0    c       1
```

and edit `/etc/conf/node.d/tp0t` so that it includes:

```
    tp0t        ntape           c       16
    tp0t        nscsitape       c       16
    tp0t        tape            c       24
    tp0t        scsitape        c       24
    tp0t        rSA/tape        c       24
```

Note that the names `scsitape` and `nscsitape` are arbitrary and can be any names that do not conflict with existing node names. Using the `kconfig` program to describe the layout of any devices will automatically create the appropriate node files.

If you are using an Exabyte drive, you must have it on-line before booting the new kernel.

8.2.8 Error Messages

`Command aborted with no apparent cause`
: This indicates an undetermined error that should be reported to your driver and/or controller supplier.

`Command timeout`
: When this error occurs on ESDI devices, it represents a hardware failure that must be investigated and repaired. On SCSI devices this may occur because of long, undisconnected operations competing on the SCSI bus, and should be retried later.

`Controller error or failure`
: This indicates a hardware failure that should be reported to your controller manufacturer.

`Correctable data error occurred`
: This indicates that the sector or block is becoming unreliable, and the data in it should be saved. On a disk device, the sector should be mapped out using the `mkpart` utility before any data is lost.

`Data address mark not found`
: This indicates a format failure on a disk. The sector should be remapped using the `mkpart` program. The data in that sector is lost.

`Data overrun`
: This is a driver error in programming a DMA operation and should be reported to your driver supplier.

`Data underrun`
: This is a driver error in programming a DMA operation and should be reported to the vendor that supplied your driver.

`Error during Format operation`
: The medium may be damaged and cannot be successfully formatted for use.

`Illegal/erroneous command`
: This indicates a driver error that should be reported to your driver supplier.

`Medium has been changed in drive`
: On a removable device such as a tape or cartridge disk, the medium has been changed without the cooperation of the driver.

`Medium is write-protected`
: On removable disks and tapes, the medium is write-protected and a write was requested.

`Sector not found`
: A request was made for a sector that does not exist (e.g., would logically be located beyond the end of the medium). This is a configuration error and should be corrected using `fdisk` or `mkpart`, as appropriate.

`(uncorrectable) Error found in sector data`
: The sector has a defect. The data is lost. The sector should be remapped using `mkpart`.

`Sector or track was marked bad`
: A sector was requested that was formatted as bad. This may occur on physical disk devices (devices with a `p` as the penultimate character in the name) when a marked-bad sector is read or written, and is a normal result. On remapped (`s`-type) disk devices, this can be avoided by using `mkpart` to remap the sector.

`Unable to recalibrate to track 0`
: A fixed disk failure has occurred. The disk needs servicing.

`Undetermined error`
: This indicates an undetermined error that should be reported to your driver and/or controller supplier.

8.2.9 See Also

Related information may be found in: *kconfig*(1), *mkpart*(1M), *mdevice*(4), *sdevice*(4), *disk*(7).

8.3 The Diskette Driver

8.3.1 Compatibility

The diskette (floppy disk) driver (also referred to as the `fd` driver, after the name of the device driver files) supports the following hardware:

Controller Chip	*Drive*
Intel 8272A, NEC 765	5 ¼" double-sided double density 5 ¼" double-sided quad density 3 ½" double-sided double density 3 ½" double-sided quad density

8.3.2 Hardware Configuration

The following display shows the hardware configuration that is supported in the standard distribution of the INTERACTIVE UNIX Operating System.

Maximum number of devices	2
Number of devices enabled	1
DMA channel	2
Interrupt priority level	4
Interrupt vector	6
Sharable interrupt	Yes
I/O address range	0x3f0–0x3f7
Controller memory address range	None

8.3.3 Software Setup

1. The `fd` driver is configured into the kernel by default. However, to reinstall the driver if it has been removed, run the `kconfig` program. Select `Drivers` under `Configure` from the `kconfig` bar menu. The system displays the drivers currently available on your machine and whether each is configured in to or out of your software.

2. To review or modify the `fd` driver, move to it and press **ENTER**. Another screen appears, describing the configuration of the driver. You can change the status of the driver (either include it or exclude it from the configuration of the next kernel you build), and you can modify individual

driver parameters. The parameters that may be modified are IRQ level, DMA channel, I/O address space, controller memory address space, and interrupt priority level.

3. To configure a driver into the kernel configuration, move to the `Driver Status` radio button and select `on`; select `off` to configure this driver out of the kernel configuration.
4. Be sure to rebuild and reinstall the kernel using `kconfig` after adding the driver.

8.3.4 Device Names

The `fd` driver provides access to diskettes as both block and raw (character) devices. For example, to access a 1.2 MB diskette as a raw device, use the device name `/dev/rdsk/f0q15dt`. However, to access the same diskette type as a block device, specify the device name `/dev/dsk/f0q15dt`. Both raw and block devices use the same major and minor numbers.

The driver controls up to two diskette drives. To access a 1.2 MB diskette on the second diskette drive as a raw device, use the device name `/dev/rdsk/f1q15dt`. The second diskette drive has the same major number as the first diskette drive. However, the minor number is 1 plus the minor number of the first diskette drive. For example, the major and minor numbers for both the first and second diskette drives supporting 1.2 MB diskettes are:

Device Name	*Major Number*	*Minor Number*
`/dev/dsk/f0q15dt`	1	0
`/dev/dsk/f1q15dt`	1	1
`/dev/rdsk/f0q15dt`	1	0
`/dev/rdsk/f1q15dt`	1	1

One diskette device configuration can have several device names. For example, `/dev/`{r}`dsk/f0q15dt`, `/dev/`{r}`dsk/f05ht`, `/dev/`{r}`fd096`, `/dev/`{r}`fd096ds15`, `/dev/`{r}`install`, and `/dev/`{r}`SA/disk0_1.2M` are different device names referring to a diskette configuration that supports 1.2 MB diskettes in the first diskette drive.

The following is a list of device names and their corresponding major and minor numbers and the supporting diskette geometry:

Device Name	*Major No.*	*Minor No.*	*Cylinder*	*Sectors Per Track*	*Sector Size*
/dev/{r}dsk/f0d4d, /dev/{r}dsk/f05d4	1	52	39	4	1024
/dev/{r}dsk/f0d4dt, /dev/{r}dsk/f05d4t	1	48	40	4	1024
/dev/{r}dsk/f0d8d, /dev/{r}dsk/f05d8	1	36	39	8	512
/dev/{r}dsk/f0d8dt, /dev/{r}dsk/f05d8t, /dev/{r}fd048ds8	1	32	40	8	512
/dev/{r}dsk/f0d9d, /dev/{r}dsk/f05d9	1	20	39	9	512
/dev/{r}dsk/f0d9dt, /dev/{r}dsk/f05d9t, /dev/{r}fd048, /dev/{r}fd048ds9, /dev/{r}SA/disk0_360k	1	16	40	9	512
/dev/{r}dsk/f0q15d, /dev/{r}dsk/f05h	1	4	79	15	512
/dev/{r}dsk/f0q15dt, /dev/{r}dsk/f05ht, /dev/{r}fd096, /dev/{r}fd096ds15, /dev/{r}install, /dev/{r}SA/disk0_1.2M	1	0	80	15	512
/dev/{r}dsk/f05d16	1	68	39	16	256
/dev/{r}dsk/f05d16t	1	64	40	16	256
/dev/{r}dsk/f05q, /dev/{r}dsk/f0q18d	1	84	79	18	512
/dev/{r}dsk/f05qt, /dev/{r}dsk/f0q18dt, /dev/{r}SA/disk0_1.44M	1	80	80	18	512
/dev/{r}dsk/f03d, /dev/{r}dsk/f0q9d	1	100	79	9	512
/dev/{r}dsk/f03dt, /dev/{r}dsk/f0q9dt, /dev/{r}fd0135ds9, /dev/{r}SA/disk0_720k	1	96	80	9	512
/dev/{r}dsk/f03h	1	116	79	16	256
/dev/{r}dsk/f03ht	1	112	80	16	256
/dev/{r}fd0	1	128	auto-sense	auto-sense	auto-sense

8.3.5 Error Messages

The `fd` driver displays three types of error messages.

- `FD(`*n*`): diskette not present - please insert`
- `FD drv` *n* `blk` *b*`: drive error message`
- `FD controller: controller error message`

The driver displays the first message at 5 second intervals if the diskette in drive *n* is removed prematurely or is not inserted quickly enough. Insert the correct diskette in the diskette drive and close the drive door.

The driver displays the second message as a drive error message. It specifies the driver number *n* and block number *b* when an error occurs after a transfer has begun. The `drive error message` may be one of the following:

`Missing data address mark`
    The diskette may not be formatted properly.

`Cylinder marked bad`
    The accessed cylinder has been marked bad during formatting.

`Seek error (wrong cylinder)`
    The drive positioned itself at the wrong cylinder when attempting to set up for the requested transfer.

`Uncorrectable data read error`
    A cyclic redundancy check (CRC) error was detected when attempting to read the requested block from the drive.

`Sector marked bad`
    The accessed sector has been marked bad during formatting.

`Missing header address mark`
    The diskette may not be formatted properly.

`Write protected`
    A write was attempted to a diskette that is currently write protected.

`Sector not found`
    The diskette may not be formatted properly.

`Data overrun`
: The system could not keep up with the requested transfer of data.

`Header read error`
: The diskette may not be formatted properly.

`Illegal sector specified`
: The driver is confused by the format of the inserted diskette.

The third type of message occurs when there is a controller error during the setup for, or actual transfer of, a block. A controller error message is one of following:

`command timeout`
: The controller failed to complete the requested command in a reasonable length of time.

`status timeout`
: The controller failed to return its status after a command was completed.

`busy`
: During an attempt to access the controller, a timeout occurred.

8.3.6 See Also

Related information may be found in: *kconfig*(1), *fd*(7).

8.4 The Asynchronous Port Driver

8.4.1 Compatibility

The asynchronous port driver (also referred to as the `asy` driver, after the name of the device driver files) supports any IBM AT-compatible serial card based on the National Semiconductor 8250, 16450, or 16550 chips.

8.4.2 Hardware Configuration

The following table describes the hardware configuration that is supported in the standard distribution of the INTERACTIVE UNIX Operating System.

Maximum number of devices	16
Interrupt priority level	7
Interrupt vector	4, 3
I/O address range	3f8–3ff, 2f8–2ff

8.4.3 *Software Setup*

The `asy` driver is configured into the kernel by default. Refer to section 10, "ADDING MODEMS, PRINTERS, AND OTHER SERIAL DEVICES," for more information.

8.4.4 *Bi-Directional Capabilities*

The INTERACTIVE UNIX System `asy` driver allows each serial port to be configured for dial-in and dial-out use at the same time. New devices are created for each port, which interlock to prevent simultaneous access to the port. This scheme negates the need for `uugetty`. Each hardware port tty0*N* (i.e., `tty01`) has the following devices associated with it:

`ttyd`*N*	Used for dial-in. `/etc/getty` should be placed on this line for incoming calls.
`acu`*N*	Used for dial-out. This is the device name that should be used when setting up the file `/usr/lib/uucp/Devices` for `cu` and `uucico`.
`tty0`*N*	Used for directly-connected devices. This is intended for hard-wired terminals, printers, and mice. It does not use the carrier-detect handshake signal and should *not* be used for modems.

8.4.4.1 Kernel Configuration. The device is configured by default for two ports. The I/O addresses and IRQs for these two ports are:

Port	*I/O Address*	*IRQ*
COM1	3f8	4
COM2	2f8	3

If you wanted to use four `asy` ports, for example, one possible way to configure `/etc/conf/sdevice.d/asy` is as follows:

```
    asy    Y    1    7    1    4    3f8    3ff    0    0
    asy    Y    1    7    1    3    2f8    2ff    0    0
    asy    Y    1    7    1    9    338    33f    0    0
    asy    Y    1    7    1    5    238    23f    0    0
```

(See *sdevice*(4) for a complete explanation of each field.)

Note the sixth, seventh, and eighth fields of the `/etc/conf/sdevice.d/asy` file, which represent the the interrupt vector (IRQ line), beginning I/O address, and ending I/O address, respectively. The second field defines whether or not that minor device should be activated in the kernel (`Y` to include). These values may be tuned for your specific serial port board. Once these extra ports are added manually, their configuration information may be changed through the `kconfig` program.

This will set up `/etc/conf/sdevice.d/asy` to configure the extra two ports to the following addresses:

Port	*I/O Address*	*IRQ*
alternate COM3	338	2*
alternate COM4	238	5

*Note that due to the AT-bus interrupt design, the IRQ 2 signal is seen by the processor as IRQ 9. As a result of this, you must configure the hardware for IRQ 2, but must configure the software to use IRQ 9. To configure the hardware, see the following section on sharable interrupts.

Use of IBM standard COM3 and COM4 ports requires that the system use sharable interrupts. The actual hardware used must also support shared interrupts. The hardware configuration used is:

Port	*I/O Address*	*IRQ*
IBM COM3	3E8	4
IBM COM4	2E8	3

8.4.4.2 Using Sharable Interrupts. Serial port boards of MCA design and certain AT serial port boards can use sharable interrupts. The INTERACTIVE UNIX System checks for possible interrupt conflicts when building a kernel. If you have a serial board that can do sharable interrupts, you must work around the `kconfig` utility. To do this:

1. Edit `/etc/conf/sdevice.d/asy` using fake interrupt values for the second-plus ports on the IRQ line (pick any unused value between 2 and 15).
2. Edit `/etc/conf/pack.d/asy/space.c`, replacing the appropriate `ASY_n_VECT` values with the actual interrupt vectors (IRQ values).

You should then generate a new kernel with the new serial port drivers. Refer to section 7 for more information about completing this procedure.

If you install more than four cards, you must edit the node file, `/etc/conf/node.d/asy`, to add the additional devices. See the *mdevice*(4) and *sdevice*(4) manual entries for details about the configuration files.

8.4.5 Device Names

The following device names are used:

Device Name	*Major Number*	*Minor Number*	*Description*
/dev/tty00	3	00	port 0 direct connect
.	.	.	.
.	.	.	.
/dev/tty03	3	03	port 3 direct connect
/dev/ttyd0	3	16	port 0 dial-in
.	.	.	.
.	.	.	.
/dev/ttyd3	3	19	port 3 dial-in
/dev/acu0	3	32	port 0 dial-out
.	.	.	.
.	.	.	.
/dev/acu3	3	35	port 3 dial-out

8.4.6 See Also

Related information may be found in: *kconfig*(1), *mdevice*(4), *sdevice*(4), *asy*(7).

8.5 The Line Printer Driver

8.5.1 Compatibility

The line printer driver (also referred to as the `lp` driver, after the name of the device driver files) supports use of a line printer on the following parallel ports:

- Monochrome adapter (printer 0, port `LPT1`)
- Parallel adapter 1 (printer 1, port `LPT2`)
- Parallel adapter 2 (printer 2, port `LPT3`)

8.5.2 Hardware Configuration

The following display shows the hardware configuration that is supported in the standard distribution of the INTERACTIVE UNIX Operating System. To support other configurations, the driver's configuration files must be modified using the `kconfig` program. If any of the configuration files are modified, the kernel must be rebuilt and reinstalled with the `kconfig` program. See *mdevice*(4) and *sdevice*(4) for details about the configuration files.

Maximum number of devices	3
Number of devices enabled	1
Interrupt priority level	3
Interrupt vector	7
LPT2 I/O address range	378 – 37f parallel port on adapter 1

Note that if more than one parallel printer port is configured into the system, each must use a separate interrupt. The following are the I/O address ranges for the other two parallel printer ports:

LPT3 I/O address range	278 – 27f parallel port on adapter 2
LPT1 I/O address range	3bc – 3bf parallel port on monochrome adapter

8.5.3 Software Setup

The `lp` driver is configured into the kernel by default. Use the `sysadm lpmgmt` utility to add a line printer to your system. Refer to section 7, "SETTING UP YOUR PRINTER," in "System Administration for New Users of the INTERACTIVE UNIX

Operating System" for information about using `sysadm` to configure line printers into your system.

8.5.4 Device Names

Device Name	*Major Number*	*Minor Number*	*Description*
`/dev/lp`	7	1	LPT1
`/dev/lp`[*0–2*]	7	0–2	LPT 1–3

If you have an `lp` port on your monochrome adapter, that port is LPT1 under DOS and `/dev/lp0` under the INTERACTIVE UNIX Operating System. The first `lp` port you have on the motherboard or on an add-in I/O adapter is LPT1 or `/dev/lp1`. However, if you have both a mono display card with parallel port *and* an add-in card, the add-in card becomes LPT2 under DOS but remains `/dev/lp1` under the INTERACTIVE UNIX Operating System.

If you have a second motherboard or add-in parallel port, it is usually LPT2 under DOS and `/dev/lp2` under the INTERACTIVE UNIX Operating System. The name varies under DOS because as DOS boots it looks for an adapter addressed one way, then another way, and then another. It calls the first parallel port it finds LPT1. Note that `/dev/lp0` corresponds to I/O port 0x3BC, `/dev/lp1` to 0x378, and `/dev/lp2` to 0x278. By default, `/dev/lp` is linked to `/dev/lp1`.

8.5.5 Tunable Parameters

The line printer driver allows users to determine how often the system checks whether a printer job is done, and how often it will send a message to the console alerting the operator that a printer needs attention.

Parameter	*Default Value*
`LP_POLLINT`	(HZ/20)
`LP_WARNINT`	(HZ*120)

where the system parameter HZ (hertz) is the number of ticks per second of the system clock (100). For example, a value of HZ*120 specifies 120 seconds.

This improves performance when using certain monochrome adapter ports. Some parallel port interfaces, such as the parallel port on many monochrome display adapters, do not latch (retain) their interrupt signals. This results in a loss of expected completion (`READY`) interrupts.

When the printer driver detects a condition requiring operator intervention (such as paper-out), it writes a message on the console. `LP_WARNINT` (default 2 minutes) defines (in minutes) the interval between these warnings.

These two parameters may be added to your system by editing `/etc/conf/pack.d/lp/space.c` to add the following:

```
# define LP_POLLINT (HZ/20)
# define LP_WARNINT (HZ*120)
```

8.5.6 Error Messages

The `lp` driver sends all error codes back to the user through the `errno` variable (see the *intro*(2) manual entry in the *INTERACTIVE SDS Guide and Programmer's Reference Manual*); no other error messages are produced.

8.5.7 See Also

Related information may be found in: *kconfig*(1), *lp*(7).

8.6 The Wangtek Cartridge Tape Driver

8.6.1 Compatibility

The Wangtek cartridge tape driver (also referred to as the `wt` driver, after the name of the device driver files) supports the following hardware:

Manufacturer	*Controller*	*Drive*
Bell Technologies	XTC-60-IC XTC-60-I XTC-125-I	Teac MT-2ST/45D (Cassette) Wangtek 5099 Wangtek 5125
COMPAQ	COMPAQ	Wangtek-compatible
Everex	EV-831, EV-833	Wangtek 5099
Wangtek	PC-36, PC-36II, 30631, 5099EK, 5150PK, 5150MK	Wangtek 5099, 512, 5150, 5150EQ, 5099EK, 5150PK, 5150MK
Cipher	5400	Wangtek-compatible
Gigatrend	Everex EV-811 Rev B	DAT Model 1236

Please note that there are three different types of cartridge tapes, and that you should not attempt to use lower density cartridges than are supported by the drive. In particular, QIC-24 (60 MB) tapes will not work properly in drives that use QIC-120 or QIC-150 (120 MB and 150 MB) tapes.

8.6.2 Hardware Configuration

The following display shows the hardware configuration that is supported in the standard distribution of the INTERACTIVE UNIX Operating System. To support other configurations, the `kconfig` program must be used (see section 8.6.3, "Software Setup").

Maximum number of devices	1
Number of devices enabled	1
DMA channel	1
Interrupt priority level	5
Interrupt vector	5
Sharable interrupt	No
I/O address range	0x300–0x301
Controller memory address range	None

Note that this driver supports only one device. You should change only the DMA channel, interrupt vector, and I/O address range.

8.6.3 Software Setup

Install the driver from the *Additional Drivers* diskette. (Refer to section 6.1 of the "INTERACTIVE UNIX Operating System Installation Instructions" for information about installing optional subsets.)

1. The `wt` driver is not configured into the kernel by default. Access the `kconfig` program to configure the driver into the kernel.

2. Select `Drivers` under `Configure` from the `kconfig` bar menu. The system displays the drivers currently available on your machine and whether each is configured in to or out of your software.

3. To review or modify the `wt` driver, move to it and press **ENTER**. Another screen appears, describing the configuration of the driver. You can change the status of the driver (either include it or exclude it from the configuration of the next kernel you build), and you can modify individual driver parameters. The parameters that may be modified are IRQ level, DMA channel, I/O address space, controller memory address space, and interrupt priority level.

4. To configure a driver into the kernel configuration, move to the `Driver Status` radio button and select `on`; select `off` to configure this driver out of the kernel configuration.

5. Be sure to rebuild and reinstall the kernel using `kconfig` after adding the driver.

8.6.4 Warnings

The following drivers provided with the INTERACTIVE UNIX Operating System are known to conflict with the default configuration of the `wt` driver:

3COM Ethernet* (`ec`)	I/O address range 0x300–0x30f
3COM 3C503 Ethernet (`el`)	I/O address range 0x300–0x30f
Future Domain TMC-830/841 SCSI host adapter (`tmc 8x0`)	Interrupt vector 5
LOGITECH* Bus Mouse (`logi`)	Interrupt vector 5
Microsoft* Bus Mouse (`mouse`)	Interrupt vector 5
Second parallel port (`lp`)	Interrupt vector 5

(Note that the INTERACTIVE UNIX Operating System is shipped with the second parallel port disabled.) If any of these drivers are to be used in the same kernel as the `wt` driver, the conflict must be resolved by rejumpering one of the boards and using `kconfig` to change the appropriate driver parameters. For example, if the second serial port is not in use, the `wt` driver and controller board can be reconfigured to use interrupt vector 3 to avoid a conflict with the LOGITECH Bus Mouse. Any board that is "soft-strapped" must be reconfigured using the setup utilities supplied by its manufacturer.

Do not use DMA channel 2 with the `wt` driver, since this DMA channel is used by the diskette drive. To make the cartridge tape stream, read or write from the drive in clusters of 256 blocks or more.

8.6.5 Device Names

Device Name	*Major No.*	*Minor No.*	*Comment*
`/dev/ntape`	28	0	No rewind on close
`/dev/rnmt0`	28	0	No rewind on close
`/dev/tape`	28	1	Rewind on close
`/dev/rmt0`	28	1	Rewind on close
`/dev/rSA/tape`	28	1	Rewind on close

8.6.6 Error Messages

`Streamer: Bad DMA channel, cannot init driver`
: The driver was configured with an illegal DMA channel number. The DMA channel must be `1` or `3`.

`Streamer: Beginning of tape`
: The beginning of the tape was encountered unexpectedly. This usually indicates an attempt to read from an uninitialized tape.

`Streamer: Block not located`
: A specific block does not exist on the tape. This error message usually results from an application or utility attempting to read a block that would be beyond the logical end-of-tape. It may also result from attempting to read an uninitialized tape. Verify that the correct tape is in the drive and that the application or utility was invoked properly.

`Streamer: Drive not online`
: The controller cannot communicate with the tape drive or the tape is broken. Check to see that the tape cartridge is not defective and recheck the hardware connections between the controller and the drive.

`Streamer: End of tape`
: The end of the tape was encountered. The usual cause is attempting to write more data to the tape than it is capable of holding.

`Streamer: Illegal command`
: The driver issued an illegal command to the controller. This can be caused by a bug in the driver or by faulty hardware.

`Streamer: No cartridge`
: No tape cartridge is in the drive, or the tape is not firmly seated in the drive. Insert a tape or reseat the tape, if one is already in the drive.

`Streamer: No data detected`
: No data was found when reading the tape. This indicates that the tape is uninitialized. Initialize the tape by writing to it before attempting to read.

`Streamer: Tape is write protected`
: An attempt was made to write to a write-protected tape. Disable the tape's write protection or use a different tape.

`Streamer: Unrecoverable data error`
Data on the tape has been damaged. This error message may also result from attempting to read an uninitialized tape. Rewrite the data to the tape, if possible, or replace the tape.

8.6.7 See Also

Related information may be found in: *kconfig*(1), *wt*(7).

8.7 The Archive Cartridge Tape Driver

8.7.1 Compatibility

The Archive cartridge tape driver (also referred to as the `ct` driver, after the name of the device driver files) supports the following hardware:

Manufacturer	*Controller*	*Drive*
Archive	VP409A	VP150i, VP150e, VP60i, VP60e, ST600

8.7.2 Hardware Configuration

The following display shows the hardware configuration that is supported in the standard distribution of the INTERACTIVE UNIX Operating System. To support other configurations, the `kconfig` program must be used (see section 8.7.3, "Software Setup").

Maximum number of devices	2
Number of devices enabled	1
DMA channel	1
Interrupt priority level	5
Interrupt vector	3
Sharable interrupt	No
I/O address range	0x200-0x201
Controller memory address range	None

8.7.3 Software Setup

1. Install the driver from the *Additional Drivers* diskette. (Refer to section 6.1 of the "INTERACTIVE UNIX Operating System Installation Instructions" for information about installing optional subsets.)

2. This driver is not configured into the kernel by default. Use the `kconfig` utility to configure the driver into the kernel. Select `Drivers` under `Configure` from the `kconfig` bar menu. The system displays the drivers currently available on your machine and whether each is configured in to or out of your software.
3. To review or modify the `ct` driver, move to it and press **ENTER**. Another screen appears, describing the configuration of the driver. You can change the status of the driver (either include it or exclude it from the configuration of the next kernel you build), and you can modify individual driver parameters. The parameters that may be modified are IRQ level, DMA channel, I/O address space, controller memory address space, and interrupt priority level.
4. To configure a driver into the kernel configuration, move to the `Driver Status` radio button and select `on`; select `off` to configure this driver out of the kernel configuration.
5. Be sure to rebuild and reinstall the kernel using `kconfig` after adding the driver.

8.7.4 Warnings

The following driver provided with the INTERACTIVE UNIX Operating System is known to conflict with the default configuration of the `ct` driver:

Serial I/O Driver (`asy`)	Interrupt vector 3

If this driver is to be used in the same kernel as the `ct` driver, the conflict must be resolved by rejumpering one of the boards and manually changing the corresponding driver configuration files. See *mdevice*(4) and *sdevice*(4) for details about the configuration files.

8.7.5 Device Names

Device Name	*Major No.*	*Minor No.*	*Comment*
`/dev/ntape`	27	0	No rewind on close
`/dev/rnmt0`	27	0	No rewind on close
`/dev/tape`	27	1	Rewind on close
`/dev/rmt0`	27	1	Rewind on close
`/dev/rSA/tape`	27	1	Rewind on close

8.7.6 Error Messages

`NO SC409A ADAPTER FOUND (getslot)`
: The driver cannot locate the controller during initialization.

`NOTICE: Cartridge Tape Controller Not Found`
: The driver cannot locate the controller during driver open time.

8.7.7 See Also

Related information may be found in: *kconfig*(1), *mdevice*(4), *sdevice*(4).

8.8 COMPAQ SCSI Tape

8.8.1 Compatibility

The COMPAQ SCSI tape driver (also referred to as the `cpqs` driver, after the name of the device driver files) provides support for the COMPAQ 320/525MB quarter-inch cartridge and associated adapter. It also supports Digital Audio Tape (DAT) units connected to the same adapter and enables hardware data compression on adapters so equipped.

8.8.2 Hardware Configuration

The following display shows the hardware configuration that is supported in the standard distribution of the INTERACTIVE UNIX Operating System. To support other configurations, the `kconfig` program must be used (see section 8.8.3, “Software Setup”).

Maximum number of devices	1
Number of devices enabled	1
DMA channel	7
Interrupt priority level	5
Interrupt vector	5
Sharable interrupt	No
I/O address range	130 – 133
Controller memory address range	None

The default hardware setup is achieved by placing all switches on the controller (SW1) in the OFF position.

8.8.3 Software Setup

1. To configure the `cpqs` driver into the kernel, access the `kconfig` program. Select `Drivers` under `Configure` from the `kconfig` bar menu. The system displays the drivers currently available on your machine and whether each is configured in to or out of your software.

2. To review or modify the `cpqs` driver, move to it and press $\boxed{\text{ENTER}}$. Another screen appears, describing the configuration of the driver. You can change the status of the driver (either include it or exclude it from the configuration of the next kernel you build), and you can modify individual driver parameters. The parameters that may be modified are IRQ level, DMA channel, I/O address space, controller memory address space, and interrupt priority level.

3. To configure a driver into the kernel configuration, move to the `Driver Status` radio button and select `on`; select `off` to configure this driver out of the kernel configuration.

4. Be sure to rebuild and reinstall the kernel using `kconfig` after adding the driver.

8.8.4 Warnings

The following drivers provided with the INTERACTIVE UNIX Operating System are known to conflict with the default configuration of the `cpqs` driver:

Future Domain TMC-830/841 SCSI host adapter (`tmc8x0`)	Interrupt vector 5
LOGITECH Bus mouse (`logi`)	Interrupt vector 5
Microsoft Bus mouse (`mouse`	Interrupt vector 5
Second parallel port (`lp`)	Interrupt vector 5

Note that the INTERACTIVE UNIX Operating System is shipped with the second parallel port disabled. If any of these drivers is to be used in the same kernel as the `cpqs` driver, the conflict must be resolved by rejumpering one of the boards and using the `kconfig` program to change the appropriate driver parameters. For example, if the second serial port is not in use, the `cpqs` driver and controller board can be reconfigured to use interrupt vector 3 to avoid a conflict with the LOGITECH Bus Mouse. Any board that is "soft-strapped" must be reconfigured using the setup utilities supplied by its manufacturer.

The 320/525 MB tape adapter may be operated at any combination of the following:

I/O address 0x330 or 0x130
Interrupt vector 3 or 5
DMA channel 5 or 7

Set the on-board switch (SW1) in accordance with the COMPAQ documentation and modify the driver's configuration files.

In addition, if the target machine is an EISA machine, the new configuration must be placed in the NonVolatile Memory by the EISA Configuration Utility *before* the new kernel is booted. Failure to run the EISA Configuration Utility at this point will cause the driver to assume the old configuration parameters in deference to the on-board switch settings. When this happens, warning messages are printed as the new kernel boots.

8.8.5 Device Names

The tape names provided by default were chosen so as not to conflict with existing tape devices and are constructed as follows for the first unit:

```
/dev/[i][n][c]rct0[-150]
```

where

i	means return immediately in when executing long-term commands such as rewind and retension.
n	means no rewind on close.
c	means enable compression.
-150	means write using QIC-150 format (512 byte blocks).

Only the first tape unit is configured. If more tapes are added, the file `/etc/conf/node.d/cpqs` must be edited manually to include new entries. Minor numbers are constructed using the following bit definitions.

Bit	*Meaning*
0x00	0=Rewind on close 1=No rewind on close
0x01	0=No compression 1=Compression enabled
0x02	0=Wait for command completion 1=Return immediately after command begins executing
0x03	0=Use tape media format and block size 1=Use QIC-150 format and block size

Once a tape has been written as a 150 MB tape, it cannot be used as a 525 MB tape until the beginning portion has been erased using the `erase` option of the `mt` command.

8.8.6 Error Messages

`Warning: Compaq SCSI Tape Adapter not configured`

There is no EISA record of the adapter (EISA machines only). Run the EISA Configuration Utility and manually add the adapter definition. The system will work but will not use the automatic configuration facilities.

`Warning: Compaq SCSI Tape EISA Config Port mismatch (Setting to 0xXX)`
: The port address for this adapter does not match the address reported by the EISA NVRAM (EISA machines only). The driver will switch to the address as reported by the EISA NVRAM. This will work unless it causes a conflict. The discrepancy should be resolved.

`Warning: Compaq SCSI Tape EISA Config IRQ mismatch (Setting to X)`
: The interrupt vector for this adapter does not match the interrupt vector reported by the EISA NVRAM (EISA machines only). The driver will switch the interrupt vector used by the adapter but will not allocate the new interrupt vector in the kernel. Under these circumstances, the tape system will not work and the discrepancy must be resolved.

`Warning: Compaq SCSI Tape EISA Config DMA mismatch (Setting to X)`
: The DMA channel for this adapter does not match the DMA channel reported by the EISA NVRAM (EISA machines only). The driver will switch to the DMA channel as reported by the EISA NVRAM. This will work unless it causes a conflict. The discrepancy should be resolved.

`Compaq SCSI Tape Adapter not responding...`
: The adapter failed to initialize.

`cpqs_scsi_reset: SCSI Adapter timed out, Status=X`
: The last command to the adapter timed out. The adapter status byte is returned.

`Adapter self test failed`
: The adapter failed to initialize.

8.8.7 See Also

Related information may be found in *kconfig*(1).

8.9 The Mini-Cartridge Floppy Tape Driver

8.9.1 Compatibility

The mini-cartridge floppy tape driver (also referred to as the `mc` driver, after the name of the device driver files) supports the following Irwin* floppy drives.

Class	*Drive Model*
110	110, 310, 410
120[XL]	120, 220, 320, 420, 720, 2020
125	125, 225, 325, 425, 725
145[XL]	145, 245, 345, 445, 745, 2040
165	165, 265, 465, 765
285XL	285, 485, 785, 2080
287XL	287, 487, 787, 2120

Note that the letters "XL" in the 120[XL] and 145[XL] may or may not be present. When XL is present, the drive is capable of servo writing extra long (i.e., 307.5 foot DC2120) tapes.

In addition, the driver supports the following controllers:

Mnemonic	*Description*
SYSFDC	System floppy controller
ALTFDC	Alternate floppy controller
4100MC	Irwin 4100MC Micro Channel controller
4100	Irwin 4100 PC Bus controller

8.9.2 Hardware Configuration

The following display shows the hardware configuration that is supported in the standard distribution of the INTERACTIVE UNIX Operating System. To support other configurations, the `kconfig` program must be used (see section 8.9.3, "Software Setup").

Maximum number of devices	2
Number of devices enabled	1
DMA channel	None
Interrupt priority level	None
Interrupt vector	None
Sharable interrupt	Yes
I/O address range	None
Controller memory address range	None

8.9.3 Software Setup

1. Install the driver from the *Additional Drivers* diskette. (Refer to section 6.1 of the "INTERACTIVE UNIX Operating System

Installation Instructions" for information about installing optional subsets.)

2. This driver is not configured into the kernel by default. Use the `kconfig` utility to configure the driver into the kernel. Select `Drivers` under `Configure` from the `kconfig` bar menu. The system displays the drivers currently available on your machine and whether each is configured in to or out of your software.
3. To review or modify the `mc` driver, move to it and press $\boxed{\text{ENTER}}$. Another screen appears, describing the configuration of the driver. You can change the status of the driver (either include it or exclude it from the configuration of the next kernel you build), and you can modify individual driver parameters. The parameters that may be modified are IRQ level, DMA channel, I/O address space, controller memory address space, and interrupt priority level.
4. To configure a driver into the kernel configuration, move to the `Driver Status` radio button and select `on`; select `off` to configure this driver out of the kernel configuration.
5. Be sure to rebuild and reinstall the kernel using `kconfig` after adding the driver.

8.9.4 Warnings

The mini-cartridge floppy tape driver does not support the no-rewind tape option.

8.9.5 Device Names

Device Name	*Major No.*	*Minor No.*	*Comment*
`/dev/mc/rmc0`	Varies	0	Mapped mode

This is the normal mode for tape read/write access. This mode recognizes tapes with either Irwin headers or older style XENIX headers in physical block 0. For both headers, defects are mapped out according to a defect list. ECC is encoded during write operations. ECC is decoded, when necessary, for read data recovery. When a Irwin header is present, (for example, AccuTrak* tapes or tapes formatted with the MC, the SCO* UNIX System, or SCO SLS drivers), the MC driver reserves the first three good blocks for header and relocation table information. The user data area starts in the fourth good tape block. A relocation table is written on the first device write for a tape that has none. When a medium error is encountered during a write operation, the block is relocated to a spare area at the end of the tape. When an older style XENIX header is present, the user data area starts in the second good tape block. No relocation table is written.

Device Name	*Major No.*	*Minor No.*	*Comment*
`/dev/mc/rmc0p`	Varies	16	Physical block mapping

This mode is intended for diagnostic use. All data (not ECC) sectors are accessible starting with the first sector in physical block 0. Defects are not mapped out. ECC is encoded for writes, and decoded, when necessary, for read data recovery.

Device Name	*Major No.*	*Minor No.*	*Comment*
`/dev/mc/rmc0p1`	Varies	32	Physical long block mapping

This mode is intended for diagnostic use. All data and ECC sectors are accessible starting with the first sector in physical tape block 0. Defects are not mapped out. ECC is not encoded for writes, but ECC is decoded, when necessary, for read data recovery.

Device Name	*Major No.*	*Minor No.*	*Comment*
`/dev/mc/rmc0a`	Varies	64	All sectors data mode

This is a compatibility mode for tape backups created with early INTERACTIVE Systems tape drivers. Since all sectors contain data starting with the first sector of physical tape block 0, there is no defect mapping and no ECC. Writing tapes with this mapping mode is discouraged.

Device Name	*Major No.*	*Minor No.*	*Comment*
`/dev/mc/rmc0nr`	Varies	80	Mapped mode with no relocation table

This device file is used for compatibility with the SCO SLS and UNIX System drivers. It is similar to map mode 0 with the exception that no relocation table is written on the first write. If, however, a relocation table is present, it is used.

Device Name	*Major No.*	*Minor No.*	*Comment*
`/dev/mc/mcdaemon`	Varies	112	Daemon special mode

This is a special file used by the daemon process.

8.9.6 See Also

Related information may be found in: *kconfig*(1), *mdevice*(4), *sdevice*(4), *mt*(1).

8.10 The Keyboard and Display Driver

8.10.1 Compatibility

The keyboard and display driver (also referred to as the `kd` driver, after the name of the device driver files) supports most IBM-compatible monochrome, CGA, EGA, and VGA adapters. In addition, most IBM-compatible keyboards (84 key, 101 key, and 102 key) are supported. The following hardware has been tested successfully:

Manufacturer	*Adapter*
COMPAQ	EGA,VGA, and Mono
Hercules*	Hercules
IBM	EGA, CGA, VGA, and Mono
Paradise	EGA, VGA
Video Seven*	Fastwrite VRAM
Bell Technologies	BLIT
Dell*	Video card, 800x600 mode
Matrox	PG Series, SG Series
Microfield*	T8*(High Resolution), V8*
Parallax	High Resolution
Pixelworks	Clipper (High Resolution) Series
Sigma Designs	LaserView* Plus

8.10.2 Hardware Configuration

The following display shows the hardware configuration that is supported in the standard distribution of the INTERACTIVE UNIX Operating System. Other configurations are not supported.

Maximum number of devices	1
Number of devices enabled	1
DMA channel	No
Interrupt priority level	6
Interrupt vector	1 (keyboard)
Sharable interrupt	No
I/O address range	0x60,0x64, 0x3b0-0x3df
Controller memory address range	0xa0000-0xbffff

8.10.3 Software Setup

The `kd` driver is configured into the kernel by default. Note that the `kconfig` program *cannot* be used to configure this driver.

8.10.3.1 Enabling Virtual Terminals. Use the command `sysadm chgvts` or access the `sysadm` program and select `Virtual Terminals` on the `Tty Management` menu under `Machine` to change the number of virtual terminals available at the console terminal. Follow the instructions on the form that appears. You may enable up to seven virtual terminals. The eighth terminal is the console terminal; it is always activated.

8.10.4 Device Names

Device Name	*Major Number*	*Minor Number*
`/dev/console`	5	0
`/dev/vt00`	5	0
`/dev/vt01`	5	32
`/dev/vt02`	5	64
`/dev/vt03`	5	96
`/dev/vt04`	5	128
`/dev/vt05`	5	160
`/dev/vt06`	5	192
`/dev/vt07`	5	224

8.10.5 Error Messages

Occasionally, while running programs such as the VP/ix Environment or INTERACTIVE X11, the system will beep when the user attempts to switch to a new virtual terminal. The user should attempt to switch again. Note that virtual terminal switching is disabled while using the kernel debugger.

8.10.6 See Also

Related information may be found in: *display*(7), *keyboard*(7).

8.11 The Keyboard Mouse Driver

8.11.1 Compatibility

The Keyboard Mouse driver (also referred to as the `kdmouse` driver, after the name of the device driver files) supports IBM-compatible keyboard mice, including the LOGITECH C9 series, when connected through a keyboard port on a Micro Channel (PS/2) or AT architecture machine.

8.11.2 Hardware Configuration

The following display shows the hardware configuration that is supported in the standard distribution of the INTERACTIVE UNIX Operating System. To support other configurations, the `kconfig` program must be used (see section 8.11.3, "Software Setup").

Maximum number of devices	1
Number of devices enabled	1
DMA channel	Not used
Interrupt priority level	6
Interrupt vector	12
Sharable interrupt	No
I/O address range	None (uses `kd`'s ports)
Controller memory address range	None

8.11.3 Software Setup

The `kdmouse` driver is not configured into the kernel by default. To install the driver, run the `kconfig` program. Select `Drivers` under `Configure` in the bar menu. Select `Keyboard Mouse` driver from the list of available drivers. When the form appears, $\boxed{\text{TAB}}$ to the `Driver Status` radio button and press the spacebar to turn the driver `On`.. Exit the form, and be sure to rebuild and reinstall the kernel to have this change take effect.

8.11.4 Device Names

Device Name	*Major No.*	*Minor No.*	*Comment*
`/dev/kdmouse`	54	0	The mouse

8.11.5 See Also

Related information may be found in: *kconfig*(1), *kdmouse*(7).

8.12 The Microsoft Bus Mouse Driver

8.12.1 Compatibility

The Microsoft Bus Mouse driver (also referred to as the `mouse` driver, after the name of the device driver files) supports all three current models of the Microsoft Bus Mouse, also referred to as the InPort Mouse.

8.12.2 Hardware Configuration

The following display shows the hardware configuration that is supported in the standard distribution of the INTERACTIVE UNIX Operating System. To support alternate configurations, the driver's configuration files must be modified and the kernel rebuilt and reinstalled using the `kconfig` program.

Maximum number of devices	1
Number of devices enabled	1
DMA channel	Not used
Interrupt priority level	6
Interrupt vector	5
Sharable interrupt	No
I/O address range	0x23c – 0x23f
Controller memory address range	None

8.12.3 Software Setup

1. Install the driver from the *Additional Drivers* diskette. (Refer to section 6.1 of the "INTERACTIVE UNIX Operating System Installation Instructions" for information about installing optional subsets.)

2. The `mouse` driver is not configured into the kernel by default. Use the `kconfig` program to configure the driver into the kernel. Select `Drivers` under `Configure` from the `kconfig` bar menu. The system displays the drivers currently available on your machine and whether each is configured in to or out of your software.

3. To review or modify the `mouse` driver, move to it and press ENTER. Another screen appears, describing the configuration of the driver. You can change the status of the driver (either include it or exclude it from the configuration of the next kernel you build), and you can modify individual driver parameters. The parameters that may be modified are

IRQ level, DMA channel, I/O address space, controller memory address space, and interrupt priority level.

4. To configure a driver into the kernel configuration, move to the `Driver Status` radio button and select `on`; select `off` to configure this driver out of the kernel configuration.
5. Be sure to rebuild and reinstall the kernel using `kconfig` after adding the driver.

No further configuration is necessary if you are using the standard hardware configuration described above.

8.12.4 Warnings

The following drivers provided with the INTERACTIVE UNIX Operating System are known to conflict with the default configuration of the `mouse` driver:

LOGITECH Bus Mouse (`logi`)	Interrupt vector 5, I/O address range 0x23c-0x23f
Wangtek Tape Drive (`wt`)	Interrupt vector 5
Future Domain TMC-830/841 SCSI host adapter (`tmc 8x0`)	Interrupt vector 5

If any of these drivers are to be used in the same kernel as the `mouse` drive, the conflict must be resolved by rejumpering one of the boards and using `kconfig` to change the corresponding driver configuration files.

For example, the I/O address range can be changed to 0x238-0x23b by jumpering the mouse card for the “Secondary InPort” setting. If a secondary serial port is not installed, the interrupt vector may be changed to IRQ 3 (4 if no serial ports are installed), and the card may also be set for IRQ 2 as long as the `mouse` driver is configured for IRQ 9 (due to the AT bus design).

8.12.5 Device Names

Device Name	*Major No.*	*Minor No.*	*Comment*
`/dev/mouse`	20	0	The mouse

8.12.6 See Also

Related information may be found in: *kconfig*(1), *mouse*(7).
The Microsoft Mouse owner's manual.

8.13 The LOGITECH Bus Mouse Driver

8.13.1 Compatibility

The LOGITECH Bus Mouse driver (also referred to as the `logi` driver, after the name of the device driver files) supports all models built to date of the LOGITECH Bus Mouse.

8.13.2 Hardware Configuration

The following display shows the hardware configuration that is supported in the standard distribution of the INTERACTIVE UNIX Operating System. To support alternate configurations, the driver's configuration files must be modified and the kernel rebuilt and reinstalled using the `kconfig` program.

Maximum number of devices	1
Number of devices enabled	1
DMA channel	Not used
Interrupt priority level	6
Interrupt vector	5
Sharable interrupt	No
I/O address range	0x23c – 0x23f
Controller memory address range	None

8.13.3 Software Setup

1. Install the driver from the *Additional Drivers* diskette. (Refer to section 6.1 of the "INTERACTIVE UNIX Operating System Installation Instructions" for information about installing optional subsets.)

2. The `logi` driver is not configured into the kernel by default. Use the `kconfig` program to configure the driver into the kernel. Select `Drivers` under `Configure` from the `kconfig` bar menu. The system displays the drivers currently available on your machine and whether each is configured in to or out of your software.

3. To review or modify the `logi` driver, move to it and press $\boxed{\text{ENTER}}$. Another screen appears, describing the configuration of the driver. You can change the status of the

driver (either include it or exclude it from the configuration of the next kernel you build), and you can modify individual driver parameters. The parameters that may be modified are IRQ level, DMA channel, I/O address space, controller memory address space, and interrupt priority level.

4. To configure a driver into the kernel configuration, move to the `Driver Status` radio button and select `on`; select `off` to configure this driver out of the kernel configuration.
5. Be sure to rebuild and reinstall the kernel using `kconfig` after adding the driver.

8.13.4 Warnings

The following drivers provided with the INTERACTIVE UNIX Operating System are known to conflict with the default configuration of the `logi` driver:

Microsoft Bus Mouse (`mouse`)	Interrupt vector 5, I/O address range 0x23c-0x23f
Wangtek Tape Drive (`wt`)	Interrupt vector 5
Future Domain TMC-830/841 SCSI host adapter (`tmc 8x0`)	Interrupt vector 5

If any of these drivers are to be used in the same kernel as the `logi` drive, the conflict must be resolved by rejumpering one of the boards and using `kconfig` to change the corresponding driver configuration files.

The I/O address on the LOGITECH mouse card cannot be changed, requiring conflicts to be resolved by changing the other card. However, if a secondary serial port is not installed, the interrupt vector may be changed to IRQ 3 (4 if no serial ports are installed), and the card may also be set for IRQ 2 as long as the `logi` driver is configured for IRQ 9 (due to the AT bus design).

8.13.5 Device Names

Device Name	*Major No.*	*Minor No.*	*Comment*
`/dev/logi`	53	0	The mouse

8.13.6 See Also

Related information may be found in: *kconfig*(1), *mouse*(7).
The LOGITECH Bus Mouse owner's manual.

9. LP PRINT SERVICE ADMINISTRATION

Note that the `sysadm` menu options for line printer administration described in section 7 of "System Administration for New Users of the INTERACTIVE UNIX Operating System" should be used for configuring standard line printer services. This section discusses the detailed workings of the LP print service system and describes how to customize the system for special printers or particular printing requirements.

9.1 Introduction

This chapter describes:

- The LP print service
- Installation of the LP print service
- Commands used to administer the system
- Stopping and starting the LP print service
- Configuring the LP print service:
    - Setting up printer configurations
    - Managing the printing load
    - Setting job priority limits for users
    - Managing pre-printed forms
    - Defining filters
- LP print service files and directories
- Writing customized filters and interface programs

9.2 How the LP Print Service Works

The LP print service, formerly known as the LP spooler, is a mechanism that allows you to send a file to be printed while you continue doing other work. The term "spool" is an acronym for "simultaneous peripheral output on-line," and LP was originally an acronym for Line Printer but has come to include many other types of printing devices. The LP print service system is software that:

- Handles the task of receiving files users want printed
- Filters the files (if needed) so they can print properly

- Schedules the work of one or more printers
- Starts programs that interface with the printer(s)
- Keeps track of the status of jobs
- Alerts you to printer problems
- Keeps track of forms currently mounted and alerts you to mount needed forms
- Issues error messages when problems arise

9.3 Summary of User Commands

The LP print service has three regular user commands:

Command	*Description*
`cancel`	Cancels a request for a file to be printed
`lp`	Sends a file or files to a printer
`lpstat`	Reports the status of the LP system

In addition to being able to send requests to the LP print service system, check the status of requests, and cancel requests, users may be given the ability to disable and enable a printer. The idea is that if a user finds a printer is malfunctioning in some way, it should not be necessary to call the administrator to turn the printer off. On the other hand, it may not be reasonable in your printing environment to allow regular users to disable a printer. You can control whether other users have access to these two commands:

Command	*Description*
`disable`	Deactivates the named printer(s)
`enable`	Activates the named printer(s)

9.4 Summary of Administrative Commands

A separate set of commands available for the LP administrator is shown in the following table. These commands are found in the `/usr/lib` directory. If you expect to use them frequently, you might find it convenient to include that directory in your `PATH` variable. To use the administrative commands, you must be logged in as either `root` or as `lp`. `lp` is a system login. (Refer to section 4 of the "INTERACTIVE UNIX Operating System Installation Instructions" for a description of how to set up a password for a system login, or use the `sysadm` program to access the `Set System Passwords` form on the `System Setup` menu under `Machine`.)

You will also probably need to use the commands for disabling and enabling a printer and the rest of the commands described in section 9.3, "Summary of User Commands."

Command	*Description*
`/usr/lib/accept`	Permits job requests to be queued for a specified destination.
`/usr/lib/reject`	Prevents jobs from being queued for a specified destination. See *accept*(1M) for more information.
`/usr/lib/lpadmin`	Sets up or changes printer configurations.
`/usr/lib/lpfilter`	Sets up or changes filter definitions.
`/usr/lib/lpforms`	Sets up or changes preprinted forms. (Use `/usr/lib/lpadmin` to mount a form.)
`/usr/lib/lpmove`	Moves output requests from one destination to another. See *lpsched*(1M) for more information.
`/usr/lib/lpsched` `/usr/lib/lpshut`	Starts the LP print service. Stops the LP print service. See *lpsched*(1M) for more information.
`/usr/lib/lpusers`	Sets or changes the default priority and priority limits the users of the LP print service can request.

The commands will now be described in the order in which they are typically used to handle the tasks needed to set up the LP print service.

9.5 Starting and Stopping the LP Print Service

Under normal operation, you should never have to start or stop the LP print service manually. It is automatically started each time the INTERACTIVE UNIX System is started and stopped each time the system is stopped. However, if you need to stop the LP print service without stopping the INTERACTIVE UNIX System as well, you can use the procedure described in the next section.

Stopping the LP print service causes all printing to cease within seconds. Any print requests that have not finished printing are printed in their entirety after the LP print service is restarted. The printer configurations, forms, and filters in effect when the LP print service is stopped are restored after it is restarted.

Jobs may pass through a printer that is not on-line. If a printer is not on-line or operating properly, you should disable the printer.

9.5.1 Manually Stopping the Print Service

To manually start and stop the LP print service, you must be logged in as either the superuser or the user `lp`. To stop the LP print service, type:

```
/usr/lib/lpshut
```

The system displays:

```
Print services stopped
```

and all printing will cease within a few seconds. If you try to stop the LP print service when it is not running, the system displays:

```
Print services already stopped
```

9.5.2 Manually Starting the Print Service

To manually restart the LP print service, type:

```
/usr/lib/lpsched
```

The system displays:

```
Print services started
```

It may take a minute or two for the printer configurations, forms, and filters to be re-established before any saved print requests start printing. If you try to restart the LP print service when it is already running, the system displays:

```
Print services already active
```

☛ The LP print service does not have to be stopped to change printer configurations or to add forms or filters.

9.6 Printer Management

Before the LP print service can start accepting print requests, you must define the configuration of each printer.

9.6.1 Defining the Configuration of a Printer

The following information can be given to define the configuration of each printer:

- Printer name
- Connection method
- Interface program
- Printer type
- Content types
- Printer port characteristics
- Character sets or print wheels
- Forms allowed
- Fault alerting
- Fault recovery
- Use restrictions
- Banner necessary
- Description
- Default printing attributes

Very little of this information is required to add a new printer to the LP print service; however, the more information you provide, the better the printer will be managed and the better it will be represented to users.

The following sections describe what the printer configuration information means and how it is used so that you can decide how to configure your printers. In each section, you will also be shown how to specify this information when adding a printer. While you can follow each of the sections in order and correctly configure a printer in several steps, you may want to wait until you have read all the sections before adding a printer so that you can do it in one step.

9.6.2 Printer Name

The printer name and the connection method (described next) are the only items required to define a new printer. The name is used to identify the printer when you change the printer configuration or manage the printer. It is also used by users who are printing files. The name may contain no more than 14 characters and can include numbers as well as letters, but no special characters other than an underscore.

You can choose any names you like, but it is good to choose names that mean something to the users of the LP print service. For example, `laser` is a good name for a laser printer, but if you have several laser printers you will want to distinguish among them, for example `laser1`, `laser2`, and `laser3`.

You do not need to fit all the descriptive information into the name since a description section exists for this purpose. You also do not have to make the name precisely identify the type of printer; users who need a particular type of printer can specify it by type, not name (see section 9.6.5, "Printer Type").

You use the printer name every time you want to refer to the printer: when adding configuration information, changing its configuration, referring to the status of the printer, and so on. The printer name can be specified by typing:

```
/usr/lib/lpadmin -p printer-name
```

☛ You must specify the connection method at the same time as the printer name (see the next section).

There are no default names; you must name every printer.

9.6.3 Connection Method

The LP print service allows you to connect your printers in a variety of ways. The simplest is to connect it directly to the computer, but you can also connect printers via a network or through a dialed modem. Once you have connected the printer to the computer, or have connected it to a network and connected the network to the computer, you should describe the connection method for the LP print service.

The default method is a direct connection. If you have used this method to connect your printer to your computer, you generally need to do only one other thing: name the connecting port. Some directly connected printers, however, can also be used as terminals for login sessions. If you want to use a printer as a terminal, you will have to arrange for the LP print service to handle it as such. To do so, use the `-l` option to the `lpadmin` command, described below.

There are two methods of making indirect connections: through a dial-up modem or over any other type of network. The LP print service uses the Basic Networking subset to handle both methods of indirect connection. When a dial-out modem is used, three prerequisites must be satisfied: the printer must be connected via a dialed modem, a dial-out modem must be connected to the computer, and the Basic Networking utilities must know about this modem.

Printers connected via any other type of network require that a *system name* be given for each printer. This is the name of an entry in the `Systems` file or related file. Although the printer is not an INTERACTIVE UNIX System, the `Systems` file can still be used to record the access method (no login information will be given, of course).

Because the `cu` program accesses a printer in the same way the LP print service does, you should set up the files as though preparing access to the printer for `cu`. The `cu` command is not used to access printers but can serve as a yardstick when setting up files: if `cu` can access a printer, the LP print service will also be able to access it. (Refer to section 11, "BASIC NETWORKING ADMINISTRATION," for details about setting up network connections.)

9.6.3.1 Adding a Directly Connected Printer. To add a directly connected printer, type:

```
/usr/lib/lpadmin -p printer-name -v path-name
```

path-name is the name of the special file representing the printer port. Typically, this is one of the following files:

```
/dev/tty00
/dev/tty01
/dev/lp
    .
    .
    .
```

9.6.3.2 Adding a Printer to be Used as a Login Terminal. To add a directly connected printer to your system for use as a login terminal:

```
/usr/lib/lpadmin -p printer-name -v pathname -l
```

As before, *pathname* is the name of the special file representing the printer port. The `-l` indicates that the printer should be automatically disabled when the LP print service is started to allow people to log in. The printer/terminal has to be manually enabled before it can be used for printing. See section 9.8, "Enabling and Disabling a Printer," for information.

9.6.3.3 Adding a Printer Connected Via a Modem or Network. To add a printer that is connected via a modem or network:

```
/usr/lib/lpadmin -p printer-name -U dial-info
```

dial-info is either the telephone number to be dialed to reach the printer's modem or the system name entered in the Basic Networking `Systems` file for the printer.

You must enter an `lpadmin` command with either the `-U` or `-v` option. And, unless you give the `-l` option, the LP print service assumes the printer is not to be used as a login terminal.

A note on dial-out or network printers: If the printer or port is busy, the LP print service will automatically retry later. This retry rate is 10 minutes if the printer is busy and 20 minutes if the port is busy. The rate is not adjustable. However, you can force an immediate retry by issuing an `enable` command for the printer. If the port or printer is likely to be busy for an extended period, you should issue a `disable` command.

The `lpstat -p` command reports the reason for a failed dial attempt. Also, if you are alerted to a dialing fault (see section 9.6.11, "Fault Alerting"), the alert message gives the reason for the fault. These messages are identical to the error messages produced by Basic Networking for similar problems.

9.6.4 Interface Program

The LP print service uses the interface program to manage the printer each time a file is printed. It has four main tasks:

- To initialize the printer port (the connection between the computer and the printer).
- To initialize the printer (restore it to a normal state in case a previously printed file has left it in an unusual state) and set the character pitch, line pitch, page size, and character set requested by the user.
- To print a banner page.
- To run a filter to print the file.

If you do not choose an interface program, the standard one provided with the LP print service is used. This should be sufficient for most of your printing needs. If you prefer, however, you can change it to suit your needs or completely rewrite your own interface program, and then specify it when you add a new printer. See section 9.19, "Customizing the Print Service," for details on how to customize an interface program.

If you will be using the standard interface program, you do not need to specify it when adding a printer. However, if you will be using a different interface program, you can either refer to it by its full path name or by referring to another printer using the same interface program.

To identify a customized interface program by name, give the printer name and the path name of the interface program as follows:

```
/usr/lib/lpadmin -p printer-name -i path-name
```

To identify a customized interface program by reference to another printer, give the printer names as follows:

```
/usr/lib/lpadmin -p printer-name1 -e printer-name2
```

printer-name$_1$ should be replaced with the name of the printer you are adding; *printer-name*$_2$ should be replaced with the name of the printer already added that is using the customized interface program.

Give the printer name and model name as follows:

```
/usr/lib/lpadmin -p printer-name -m model-name
```

to identify an interface program by reference to a model interface program.

9.6.5 Printer Type

The *printer-type* is important for the proper use of the printer. The LP print service uses the *printer-type* to extract information about the printer from the Terminfo database. This information describes the capabilities of the printer so that you can be warned if some of the configuration information you provide is not appropriate for the printer. The information also describes the control data to use to initialize the printer before printing a file. While you are not required to specify a *printer-type*, you are urged to specify one so that better print services will be provided.

The *printer-type* is the generic name for the printer. Typically it is derived from the manufacturer's name, such as `495` for the AT&T 495 Laser Printer. Appendix F in the *User's Guide* provides a description of how to determine a correct `TERM` variable for a user terminal and can be used as a guide for picking an acceptable name for your printer.

Specify the *printer-type* as follows:

```
/usr/lib/lpadmin -p printer-name -T printer-type
```

If you do not define the *printer-type*, the default `unknown` is used. This produces empty results when the LP print service looks up information about the printer, so the print service is not able to verify certain requests or initialize the printer. The options for page width, page length, characters per inch (cpi), and lines per inch (lpi) cannot be set unless the Terminfo entry supports these options. See section 9.6.16, "Default Printing Attributes," for additional information on default settings.

9.6.6 Content Types

The *printer-type* information tells the LP print service what type of printer is being added; the content-type information tells the LP print service what types of files can be printed. Most printers can print only one type of file; for these, the content-type is likely to be identical to the *printer-type*. However, some printers, can accept several different types of files and print their contents properly.

When adding this kind of printer, you should list the names of the content types it accepts.

When a file is submitted to the LP print service for printing, the print service searches for a printer capable of handling the job. The print service can identify an appropriate printer through either the content-type name or the *printer-type* name. Therefore, you may specify either name (or no name) when submitting a file for printing.

Content-type names may look a lot like *printer-type* names, but you are free to choose names that mean something to you and the people using the printer. (The names `simple`, `terminfo`, and `any` are recognized as having particular meanings by the LP print service; be sure to use them consistently.) The names must contain no more than fourteen characters and may include only letters, digits, and underscores. If the same content type is printable by several different types of printers, you should use the same content-type names when you add those printers. This makes it easier for the people using the printers because they can use the same name to identify the type of file they want printed regardless of the printing destination.

For example, several manufacturers produce printers that accept PostScript files. While these printers may need different printer types so that each can be properly initialized (assuming the initialization control sequences are different), they may all be capable of handling the same type of input file, which you may call, for example, `postscript`. As another example, several manufacturers produce printers that accept ANSI X3.64-defined escape sequences. However, the printers may not support all the ANSI capabilities or may support different sets of capabilities. You may want to give different content-type names for these printers to differentiate them.

You do not have to list the content types for a printer. If you do not, the *printer-type* is used as the name of the content type the printer can handle. If you have not specified a *printer-type*, the LP print service assumes the printer can print only files of content type `simple`. This may be sufficient if you require people to pick the proper printer and make sure the files are properly prepared for the printer before they are submitted for printing.

One type of file often encountered on the INTERACTIVE UNIX System is called `simple`. This file is assumed to contain only printable ASCII characters and the following control characters:

backspace
: moves the carriage back one space except at the beginning of a line

tab
: moves the carriage to the next tab stop, which is normally every eight columns on most printers

linefeed
: moves the carriage to the beginning of the next line (may require special port settings for some printers. Refer to section 9.6.7, "Printer Port Characteristics.")

form feed
: moves the carriage to the beginning of the next page

carriage return
: moves the carriage to the beginning of the same line (may fail on some printers)

The word "carriage" may be archaic for modern laser printers, but similar actions apply. If a printer can handle a `simple` file, you should include it in the content type list when you add the printer and specify the content type(s) the printer can handle. If you do not want a printer to accept files of type `simple`, you must give an alternate list of content types the printer can accept. (The *printer-type* is a good name to use if no other type is appropriate.)

Another content-type name is `terminfo`. This does not refer to a particular type of file but instead refers to all the types represented in the Terminfo database. It is not likely that any printer is capable of handling all the types listed in the database. However, this name is reserved for describing possible filter capabilities. Likewise, the content type `any` is reserved for describing the types of files a filter can accept or produce. These names should not be used as content types when adding a printer.

Specify the list of content types as follows:

```
/usr/lib/lpadmin -p printer-name -I content-type-list
```

The *content-type-list* is a list of names separated by a comma or space. If you use spaces to separate the names, enclose the entire list (but not the `-I`) in quotes. If you do not define the types of files a printer can accept, the LP print service assumes it can take type `simple` and a type with the same name as the *printer-type* (if the *printer-type* is defined).

9.6.7 Printer Port Characteristics

Printers connected directly to computers and those connected over some networks require that the printer port characteristics be set by the interface program. These characteristics define the low-level communications with the printer. Included are the baud rate; use of XON/XOFF flow control; 7, 8, or other bits per byte; style of parity; and output post-processing. The standard interface program uses the `stty` command to initialize the printer port, minimally setting the baud rate and a few other default characteristics.

The default characteristics applied by the standard interface program are listed below.

Default	*Meaning*
`9600`	9600 baud rate
`cs8`	8-bit bytes
`-cstopb`	1 stop bit per byte
`-parenb`	No parity generation
`ixon`	Enable XON/XOFF flow control
`-ixany`	Allow only XON to restart output
`opost`	Post-process data stream as listed below:
`-oluc`	Do not map lowercase to uppercase
`onlcr`	Map linefeed into carriage-return/linefeed
`-ocrnl`	Do not map carriage-return into linefeed
`-nocr`	Output carriage-returns even at column 0
`nl0`	No delay after linefeeds
`cr0`	No delay after carriage-returns
`tab0`	No delay after tabs
`bs0`	No delay after backspaces
`vt0`	No delay after vertical tabs
`ff0`	No delay after form-feeds

You may find that the default characteristics are sufficient for your printers. However, printers vary enough that you are likely to find that you have to set different characteristics. See *stty*(1).

If you have a printer that requires printer port characteristics other than those handled by the `stty` program, you will have to customize the interface program. See section 9.19, "Customizing the Print Service," for more information.

When you add a new printer, you can specify an additional list of port characteristics that should be applied when printing each user's file. The list you give is applied after the default list so that you do not need to include in your list default items that you do not want to change. Specify the additional list as follows:

```
/usr/lib/lpadmin -p printer-name -o "stty='stty-option-list'"
```

Note that both the double quotes and single quotes are needed if you give more than one item in the *stty-option-list*. If you do not include alternate printer port characteristics, the default list in the table is used.

As one example, suppose your printer is to be used for printing graphical data, where linefeed characters should be output alone without an added carriage-return. Enter the following command:

```
/usr/lib/lpadmin -p printer-name -o "stty=-onlcr"
```

Note that the single quotes are omitted because there is only one item in the list.

As another example, suppose your printer requires odd parity for data sent to it. Enter the following command:

```
/usr/lib/lpadmin -p printer-name -o "stty='parenb parodd cs7'"
```

9.6.8 Character Sets or Print Wheels

Printers differ in the way they can print in different font styles. Some have changeable print wheels, some have changeable font cartridges, others have preprogrammed, selectable character sets. The LP print service, with your help, can minimize the impact of these differences on the users of the LP print service.

When adding a printer, you can specify what print wheels, font cartridges, or character sets are available with the printer. Only one of these is assumed to apply to each printer. From the point of view of the LP print service, however, print wheels and changeable font cartridges are the same because they require you to intervene and mount a new print wheel or font cartridge. Thus, for ease of discussion, only print wheels and character sets will be mentioned.

When you list the print wheels or character sets available, you will be assigning names to them. These names are for your convenience and the convenience of the users. Because different printers may have similar print wheels or character sets, you should use common names for all printers. This allows a person to submit a file for printing and ask for a particular font style, without regard for which printer will be used or whether a print wheel or selectable character set is used.

If the printer has mountable print wheels, you need only list their names. If the printer has selectable character sets, you need to list their names and map each one into a name or number that uniquely identifies it in the Terminfo database. If you are using 386/ix* Operating System Release 2.0 or later or version 2.2 or later of the INTERACTIVE UNIX Operating System, you can use the following command to determine the names of the character sets listed in the Terminfo database:

```
TERM=printer-type tput csnm 0
```

printer-type is the name of the printer type in question. The name of the 0th character set (the character set obtained by default after the printer is initialized) should be printed. Repeat the command, using 1, 2, 3, and so on in place of the 0, to see the names of the other character sets. In general, the Terminfo names should closely match the names used in the user documentation for the printer. However, since some manufacturers use different names, the Terminfo names may differ from one printer type to the next.

☛ For the LP print service to be able to find the names in the Terminfo database, you must specify a printer type. See section 9.6.5, "Printer Type," for more information.

To specify a list of print wheel names when adding a printer, type:

```
/usr/lib/lpadmin -p printer-name -S print-wheel-list
```

print-wheel-list is a list of names separated by a comma or space. If you use spaces to separate the names, enclose the entire list (but not the -S) in quotes.

To specify a list of character set names and to map them into Terminfo names or numbers, type:

```
/usr/lib/lpadmin -p printer-name -S character-set-list
```

character-set-list is also a list of names separated by a comma or space; however, each item in the list looks like one of the following:

```
csN=character-set-name
character-set-name1=character-set-name2
```

N in the first case is a number from 0 to 63 that identifies the number of the character set in the Terminfo database. *character-set-name*$_1$ in the second case identifies the character set by its Terminfo name. In either case, the name to the right of the equal sign (=) is the name you choose as an alias of the character set.

☛ You do not have to provide a list of aliases for the character sets if the Terminfo names are adequate. You can refer to a character set by number, by Terminfo name, or by your alias.

For example, suppose your printer has two selectable character sets (sets #1 and #2) in addition to the standard character set (set #0). The printer type is `5310`. Type the following to determine the names of the selectable character sets:

```
TERM=5310 tput csnm 1
english
TERM=5310 tput csnm 2
finnish
```

The words `english` and `finnish`, the output of the commands, are the names of the selectable character sets. Suppose that the name `finnish` is adequate for referring to character set #2, but better names are needed for the standard set and set #1. Type the following command to define synonyms:

```
/usr/lib/lpadmin -p printer-name -S "cs0=american, english=british"
```

If you do not list the print wheels or character sets that can be used with a printer, then the LP print service assumes the following: a printer that takes print wheels has only a single, fixed print wheel; people cannot ask for a special print wheel when using the printer; and a printer that has selectable character sets can take any `cs`*N* name or Terminfo name known for the printer.

9.6.9 Alerting to Mount a Print Wheel

If you have printers that can take changeable print wheels and you have listed the print wheels allowed on each, then users will be able to submit a print request to use a particular print wheel. However, until it is mounted (refer to section 9.6.19, "Mounting a Form or Print Wheel"), a request for a print wheel will stay queued and will not be printed. You could periodically monitor the number of print requests pending for a particular print wheel, but the LP print service provides an easier way. You can ask to be alerted when the number of requests waiting for a print wheel has exceeded some threshold.

You can choose any of several ways to receive an alert:

- You can receive an alert via electronic mail. See *mail*(1).
- You can receive an alert written to whatever terminal on which you are logged in. See *write*(1).
- You can receive an alert through a program of your choice.
- You can receive no alerts.

☛ If you elect to receive no alerts, you must check to see whether the proper print wheel is mounted.

In addition to the method of alerting, you can also set the number of requests that must be queued before you are alerted, and you can arrange for repeated alerts every few minutes until the print wheel is mounted. You can choose the rate of repeated alerts, or you can choose to receive only one alert per print wheel.

To arrange for alerting to the need to mount a print wheel, type one of the following commands:

```
/usr/lib/lpadmin -S print-wheel-name -A mail -Q integer -W minutes
/usr/lib/lpadmin -S print-wheel-name -A write -Q integer -W minutes
/usr/lib/lpadmin -S print-wheel-name -A 'command' -Q integer -W minutes
/usr/lib/lpadmin -S print-wheel-name -A none
```

The first two commands direct the LP print service to send you a mail message or write the message directly to your terminal, respectively, for each alert. The third command directs the LP print service to run *command* for each alert. The shell environment currently in effect when you enter the third command is saved and restored for the execution of *command*; this includes the environment variables, user and group IDs, and current directory. The fourth command above directs the LP print service to never send you an alert when the print wheel needs to be mounted. *integer* is the number of requests that need to be waiting for the print wheel, and *minutes* is the number of minutes between repeated alerts.

☛ If you want mail sent or a message written to another person when a printer fault occurs, use the third command listed. Use the options:

```
    -A 'mail user-name'
```

or

```
    -A 'write user-name'
```

Once you start receiving repeated alerts, you can direct the LP print service to stop sending you alerts for the current case by typing:

```
/usr/lib/lpadmin -S print-wheel-name -A quiet
```

Once the print wheel has been mounted and unmounted again, alerts will start again if too many requests are waiting. Alerts will also start again if the number of requests waiting falls below the `-Q` threshold and then rises up to the `-Q` threshold again, as when waiting requests are canceled or if the type of alerting is changed.

If *print-wheel-name* is `all` in any of the commands above, the alerting condition applies to all print wheels for which an alert has already been defined.

If you do not define an alert method for a print wheel, you will not receive an alert for it. If you do define a method but do not give the `-W` option, you will be alerted once for each occasion.

9.6.10 Forms Allowed

For a description of forms, see section 9.13, "Forms."

You can limit the use of preprinted forms on any printer. You may want to do this, for example, if a printer is not well suited for printing on a particular form because of low print quality or if the form cannot be lined up properly in the printer.

The LP print service uses the list of forms allowed or denied for a printer to warn you against mounting a denied form on the printer. However, you have the final word on this; the LP print service will not refuse such an attempt. The LP print service will, however, refuse a user's request to print a file on a printer using a form denied on that printer unless the form is already mounted.

If you try to list a form as allowed on a printer but the printer does not have sufficient capabilities to handle the form, the command will be rejected.

The method of listing the forms allowed or denied for a printer is similar to the method used to list those users allowed or denied access to the `cron` and `at` facilities. See *crontab*(1). Briefly, the rules are as follows:

- An allow list is a list of forms that you are allowed to use with the printer. A deny list is a list of forms for which you are denied permission to use with the printer.
- If the allow list is not empty, the deny list is ignored. If the allow list is empty, the deny list is used. If both lists are empty, there are no restrictions on which forms can be used.
- Putting `any` or `all` into the allow list allows all forms; putting `any` or `all` into the deny list denies all forms.

You can add names of forms to either list using one of the following commands:

```
/usr/lib/lpadmin -p printer-name -f allow:form-list
/usr/lib/lpadmin -p printer-name -f deny:form-list
```

form-list is list of names of forms separated by a comma or space. If you use spaces to separate names, enclose the entire list (including `allow:` or `deny:` but not `-f`) in quotes. The first command adds names to the allow list and removes them from the deny list. The second command adds names to the deny list and removes them from the allow list. To make use of all permissible forms, specify `allow:all`; to deny permission for all forms, specify `deny:all`.

If you do not add forms to the allow list or deny list, the LP print service considers that the printer denies the use of all forms. It will, however, allow you to mount any form. It also provides a warning message if the form is not in the allow list or if you are attempting to mount a form that does not match the capabilities of the printer as described earlier.

9.6.11 Fault Alerting

The LP print service provides a framework for detecting printer faults and alerting you. Faults can range from simple problems, such as running out of paper or ribbon or needing to replace the toner, to more serious faults, such as a local power failure or printer failure. The range of fault indicators is also broad, ranging from dropping carrier (the signal that indicates that the printer is online), to sending an XOFF, to sending a message. Only two classes of printer fault indicators are recognized by the LP print service itself: a drop in carrier and an XOFF not followed within a reasonable time by an XON. However, you can add filters that can recognize any other printer fault indicators and rely on the LP print service to alert you to a fault when the filter detects it.

☛ For a description of how to add a filter, refer to section 9.14, "Filter Management." For a description of how a filter should let the LP print service know a fault has occurred, refer to section 9.19, "Customizing the Print Service."

You can choose one of several ways to receive an alert to a printer fault:

- You can receive an alert via electronic mail. See *mail*(1).
- You can receive an alert written to the terminal on which you are logged in (any terminal). See *write*(1).
- You can receive an alert through a program of your choice.
- You can receive no alerts.

☛ If you elect to receive no alerts, you will need a way of finding out about the faults and fixing them; the LP print service will not continue to use a printer that has a fault.

In addition to the method of alerting, you can also arrange for repeated alerts every few minutes until the fault is cleared. You can choose the rate of repeated alerts, or you can choose to receive only one alert per fault.

☛ Without a filter that provides better fault detection, the LP print service cannot automatically determine when a fault has been cleared except by trying to print another file. It assumes that a fault has been cleared when it is successfully able to print a file. Until that time, if you have asked for only one alert per fault, you will not receive another alert. If the printer faults again, after you have fixed a fault but before the LP print service has tried printing another file, or if your attempt to fix the fault did not succeed, you are not notified. Receiving repeated alerts per fault or requiring manual re-enabling of the printer will overcome this problem (refer to section 9.6.12, "Fault Recovery").

To arrange for alerting to a printer fault, type one of the following commands:

```
/usr/lib/lpadmin -p printer-name -A mail -W minutes
/usr/lib/lpadmin -p printer-name -A write -W minutes
/usr/lib/lpadmin -p printer-name -A 'command' -W minutes
/usr/lib/lpadmin -p printer-name -A none
```

The first two commands direct the LP print service to send you a mail message or write the message directly to your terminal, respectively, for each alert. The third command directs the LP print service to run *command* for each alert. The shell environment currently in effect when you enter the third command is saved and restored for the execution of *command*. The environment includes environment variables, user and group IDs, and current directory. The *minutes* is the number of minutes between repeated alerts. The fourth command above directs the LP print service to not send you an alert when a fault occurs.

☛ If you want mail sent or a message written to another person when a printer fault occurs, use the third command. Use the option

```
    -A 'mail user-name'
```

or

```
    -A 'write user-name'
```

Once a fault occurs and you start receiving repeated alerts, you can direct the LP print service to stop sending you alerts for the current fault by typing:

```
/usr/lib/lpadmin -p printer-name -A quiet
```

If *printer-name* is `all` in any of the commands above, the alerting condition applies to all printers.

If you do not define an alert method, you will receive mail once for each printer fault. If you do define a method but do not give the `-W` option, you will be alerted once for each fault.

9.6.12 Fault Recovery

Once a printer fault has been detected and you have been alerted, you will probably fix the fault and get the printer ready for printing. When the printer is ready for printing again, the LP print service recovers in one of three ways:

- Continues printing at the top of the page where printing stopped.
- Restarts printing at the beginning of the print request that was active when the fault occurred.
- Waits for you to tell the LP print service to re-enable the printer.

☛ The ability to continue printing at the top of the page where printing stopped requires the use of a filter that can wait for a printer fault to be cleared before resuming properly. Such a filter probably has to have detailed knowledge of the control sequences used by the printer so it can keep track of page boundaries and know where in a file printing stopped. The default filter used by the LP print service cannot do this. If a proper filter is not being used, you will be notified in an alert if recovery cannot proceed as you want.

To specify the way the LP print service should recover after a fault has been cleared, type one of the following commands:

```
/usr/lib/lpadmin -p printer-name -F continue
/usr/lib/lpadmin -p printer-name -F beginning
/usr/lib/lpadmin -p printer-name -F wait
```

These direct the LP print service, respectively, to continue at the top of the page, restart from the beginning, or wait for you to enter an `enable` command to re-enable the printer (refer to section 9.8, "Enabling and Disabling a Printer," for information on the `enable` command).

If you do not specify how the LP print service is to resume after a printer fault, it will try to continue at the top of the page where printing stopped, or failing that, at the beginning of the print request.

If the recovery is `continue` but the interface program does not continue to run so that it can detect when the printer fault has been cleared, printing will be attempted every few minutes until it succeeds. You can force the LP print service to retry immediately by issuing an `enable` command.

9.6.13 Restricting User Access

You can limit the use of a printer to a subset of all people on your computer. You may want to do this, for example, if a printer is being set aside for printing sensitive information and only a subset of the people can print sensitive information or if use of a high quality printer incurs expenses not all people are allowed to incur.

The LP print service uses the list of users allowed or denied for a printer to restrict use of the printer. The LP print service refuses a user's request to print a file on a printer he or she is not allowed to use.

The method of listing the users allowed or denied for a printer is similar to the method used to list users allowed or denied access to the `cron` and `at` facilities and the method described in section 9.6.10, "Forms Allowed." Briefly, the rules are as follows:

- An allow list is a list of those users allowed to use the printer. A deny list is a list of those users denied access to the printer.
- If the allow list is not empty, the deny list is ignored. If the allow list is empty, the deny list is used. If both lists are empty, there are no restrictions on who can use the printer.
- Putting `any` or `all` into the allow list allows everybody to use the printer; putting `any` or `all` into the deny list denies everybody, except the `lp` user and the superuser.

You can add names of users to either list using either of the following:

```
/usr/lib/lpadmin -p printer-name -u allow:user-list
/usr/lib/lpadmin -p printer-name -u deny:user-list
```

user-list is a list of names of users separated by a comma or space. If you use spaces to separate the names, enclose the entire list (including `allow:` or `deny:` but not the `-u`) in quotes. The first command adds the names to the allow list and removes them from the deny list. The second command adds the names to the deny list and removes them from the allow list. Using `allow:all` allows everyone; using `deny:all` denies everyone.

If you do not add user names to the allow or deny lists, the LP print service assumes that everyone can use the printer.

9.6.14 Banner Necessary

Usually you will want to have each print request preceded with a banner page. The banner page shows who requested the printing, the request ID, and when it was printed, and allows for an optional title that the requester can use to better identify the printout. The banner page greatly eases the task of separating a sequence of print requests so that each can be given to the correct user or placed in separate bins.

Sometimes a user needs to avoid printing a banner page. The likely occasions are when the printer has forms mounted that should not be wasted, such as payroll checks or accounts payable checks. Printing a banner page in such occasions may cause problems.

To allow a user to skip the banner page, type:

```
/usr/lib/lpadmin -p printer-name -o nobanner
```

If you later change your mind, you can remove this choice by typing:

```
/usr/lib/lpadmin -p printer-name -o banner
```

If you do not allow a user to skip the banner page, the LP print service rejects all attempts to avoid a banner page when printing on the printer. This is the default action.

9.6.15 Description

You can add a description of a printer that can give people using the LP print service helpful information about the printer. This description can contain any message you like, including a room number where the printer is found, who to call with printer problems, and so on.

Use:

```
lpstat -D -p printer-name
```

to see the message.

To add a description when adding a printer, type:

```
/usr/lib/lpadmin -p printer-name -D 'text'
```

text is the message. Include the quotes if the message contains blanks or other characters that the shell might interpret if the quotes are left out.

Unless you give a printer description, none will be presented to people who ask about it.

9.6.16 Default Printing Attributes

When a user submits a request to print a file, the page size, character pitch, and line pitch (i.e., print spacing) are normally determined from the form that will be printed on. If the user does not require a form, he or she can give the page size and print spacing to use. However, if he or she gives neither a form to use nor the page size and print spacing, defaults are used.

You can set the defaults for each printer. This can also serve to make submitting a print request easier, by designating different printers as having different default page sizes or print spacing. Users then simply route their file to the appropriate printer to get the style output they want. For example, you can have one printer dedicated to printing wide (132 column) output, another printing normal (80 column by 66 lines) output, yet another printing letter quality (12 characters per inch, 8 lines per inch).

You can independently specify four default settings: page width, page length, character pitch, and line pitch. You can scale these to fit your needs. The first two can be given in columns and lines, inches, or centimeters. The last two can be given as characters and lines per inch or per centimeter. In addition, the character pitch can be specified as `pica` for 10 cpi, `elite` for 12 cpi, or `compressed` for the maximum cpi the printer can provide (up to a limit of 30 cpi).

Set the defaults using one or more of the following commands:

```
/usr/lib/lpadmin -p printer-name -o width=scaled-number
/usr/lib/lpadmin -p printer-name -o length=scaled-number
/usr/lib/lpadmin -p printer-name -o cpi=scaled-number
/usr/lib/lpadmin -p printer-name -o lpi=scaled-number
```

Add the letter `i` to *scaled-number* to indicate inches, or the letter `c` to indicate centimeters. The letter `i` for character pitch (cpi) or line pitch (lpi) is redundant. You can also give `pica`, `elite`, or `compressed` instead of a number for the character pitch.

If you do not provide defaults, the page size and print spacing are those available when the printer is initialized. You can find out what the defaults are by first defining the printer configuration without providing your own defaults, then using the `lpstat` program to display the printer configuration. The command:

```
lpstat -p printer-name -l
```

reports the default page size and print spacing. If you have not provided the defaults, the reported defaults are calculated from the Terminfo database entry for the printer. Obviously, this requires you to have provided a printer type in the printer configuration.

9.6.17 Adding a Printer to a Class

It is occasionally convenient to treat a collection of printers as a single class. The benefit is that a person can submit a file for printing by a member of a class, and the LP print service picks the first printer in the class that it finds free. This allows faster turnaround, as printers are kept as busy as possible.

Classes are not needed if the only purpose is to allow a user to submit a print request by type of printer. The `lp -T` *type* command allows a user to submit a file and specify its type. The first available printer that can handle the type of file is used to print the file. The LP print service avoids using a filter, if possible, by choosing a printer that can print the file directly over one that would need it filtered first. Refer to section 9.14, "Filter Management," for more information about filters.

Classes do have uses, however. One use is to put into a class a series of printers that should be used in a particular order. If you have a high-speed printer and a low-speed printer, for example, you probably want the high-speed printer to handle as many print requests as possible, with the low-speed printer reserved for use when the other is busy. Because the LP print service always checks for an available printer in the order the printers were added to a class, you could add the high-speed printer to the class before the low-speed printer and let the LP print service route print requests in the order you wanted.

Add a printer to a class using the following command:

```
/usr/lib/lpadmin -p printer-name -c class-name
```

If the class *class-name* does not exist yet, it will be created.

☛ Class names and printer names must be unique. This allows a user to specify the destination for a print request without having to know whether it is a class of printers or a single printer. Thus, you cannot have a class and printer with the same name.

Until you add a printer to a class, it will not belong to any class.

9.6.18 Setting the System Default Destination

You can define the printer or class to be used to print a file when the user has not explicitly asked for a particular destination and has not set the LPDEST shell variable. The printer or class must already be in existence.

Make a printer or class the default destination by typing:

```
/usr/lib/lpadmin -d printer-or-class-name
```

If you later decide that there should be no default destination, enter a null *printer-or-class-name* as in the following command:

```
/usr/lib/lpadmin -d
```

If you do not set a default destination, there will be none. Users will have to explicitly name a printer or class in each print request, or will have to set the LPDEST shell variable with the name of a destination.

9.6.19 Mounting a Form or Print Wheel

Refer to section 9.13, "Forms," for information about preprinted forms. Before the LP print service will start printing files that need a pre-printed form or print wheel, you will have to mount it on a printer. If alerting has been set on the form or print wheel, you will be alerted when enough print requests are queued waiting for it to be mounted.

When you mount a form, you may want to see if it is lined up properly. If an alignment pattern has been registered with the form, you can ask that this be repeatedly printed after you have mounted the form, until you have adjusted the printer so that the alignment pattern appears to be correct.

Mounting a form or print wheel involves first loading it onto the printer and then telling the LP print service that it is mounted. Because it is difficult to do this on a printer that is currently printing and because the LP print service will continue to print files not needing the form on the printer, you will probably have to disable the printer first. The proper procedure is to:

1. Disable the printer using the `disable` command.
2. Mount the new form or print wheel as described below.
3. Re-enable the printer using the `enable` command. (The `disable` and `enable` commands are described in section 9.8, "Enabling and Disabling a Printer.")

When you have loaded the new form or print wheel into the printer, type the following command to tell the LP print service to mount it. (This command is shown on two lines for readability; it must be entered as one line.)

```
/usr/lib/lpadmin -p printer-name -M -S print-wheel-name
                 -f form-name -a -o filebreak
```

Leave out `-S` *print-wheel-name* if you are mounting just a form, or leave out the `-f` *form-name* `-a -o filebreak` if you are mounting just a print wheel.

If you are mounting a form, you will be asked to press ENTER before each copy of the alignment pattern is printed. After the pattern is printed, you can adjust the printer and press ENTER again. If no alignment pattern has been registered, you will not be asked to press ENTER. You can drop the `-a` and `-o filebreak` options if you do not want to bother with the alignment pattern.

The `-o filebreak` option tells the LP print service to add a "formfeed" after each copy of the alignment pattern. The actual control sequence used for the formfeed depends on the printer involved and is obtained from the Terminfo database. If the alignment pattern already includes a formfeed, leave out the `-o filebreak` option.

If you want to unmount a form or print wheel, use the following command:

```
/usr/lib/lpadmin -p printer-name -M -S none -f none
```

Leave out `-S none` if you just want to unmount a form; likewise, leave out `-f none` if you just want to unmount a print wheel.

Until you have mounted a form on a printer, only print requests that do not require a form will be sent to it. Likewise, until you have mounted a print wheel on a printer, only print requests that do not require a particular print wheel will be sent to it.

9.6.20 Removing a Printer or Class

You can remove a printer or class if it has no pending print requests. If there are pending requests, you must first move them to another printer or class using the `lpmove` command, or remove them using the `cancel` command.

Removing the last remaining printer of a class automatically removes the class as well. However, the removal of a class does not cause the removal of printers that were members of the class. If the printer or class removed is also the system default destination, the system will no longer have a default destination.

To remove a printer or class, type:

```
/usr/lib/lpadmin -x printer-or-class-name
```

If all you want to do is remove a printer from a class but not delete the printer, type:

```
/usr/lib/lpadmin -p printer-name -r class-name
```

This leaves the printer intact.

9.6.21 Putting It All Together

When adding a new printer, you can do it in separate steps, entering the commands described above. However, you may find it easier to enter one or two commands that combines all the necessary arguments. Here are some examples.

9.6.21.1 Example 1. Add a new printer called `lp1` on printer port `/dev/tty00`. It should use the standard interface program, with the default page size of 90 columns by 71 lines, and linefeeds should not be mapped into carriage return/linefeed pairs.

```
/usr/lib/lpadmin -p lp1 -v /dev/tty00 -T 455 -o
"width=90 length=71 stty=-onlcr"
```

(The preceding line is split into two lines for readability in this document. It must be typed as one line.)

9.6.21.2 Example 2. Add a new printer called `laser` on printer port `/dev/tty01`. It should use a customized interface program, it can handle three file types: `i10`, `i300`, and `impress`, and only the users `doceng` and `docpub` may use it.

```
/usr/lib/lpadmin -p laser -v /dev/tty01 -i /usr/doceng
/laser_intface -I "i10,i300,impress" -u "allow:doceng,docpub"
```

(The preceding line is split into two lines for readability in this document. It must be typed as one line.)

9.6.21.3 Example 3. No alerting was set when adding the `lp1` printer in the first example. If you want to add alerting every 10 minutes after a fault until the problem is fixed, type:

```
/usr/lib/lpadmin -p lp1 -A write -W 10
```

9.7 Accepting Print Requests for a New Printer

Initially, the LP print service will not consider a new printer eligible for printing files. This gives you time to make sure you have defined the printer configuration the way you want. When you are ready to make the printer available to others, you have to tell the LP print service.

There are two steps in making a printer ready for use after you have defined the printer configuration. First, the LP print service has to be told to accept print requests for the new printer. Second, the new printer has to be enabled to print. These are separate tasks because you may have occasion to want to do one but not the other.

Telling the LP print service to accept print requests for the new printer is done with the `accept` command. These is more about this command in section 9.11, "Managing the Printing Load." For now, all you need to know is that you should enter the following command to allow this printer to be used:

```
/usr/lib/accept printer-or-class-name
```

As you can see, this command is also needed to let the LP print service start accepting print requests for a class.

9.8 Enabling and Disabling a Printer

When a printer is ready for use and the LP print service is accepting print requests for it, you will have to enable it before anything will print. You will use the `enable` command to do this. Having the LP print service wait for you instead of automatically starting to print files lets you make sure that the correct form is loaded in the printer, that the correct print wheel or font cartridge is in place, and that the printer is on-line.

When all is ready, type the following command to enable printing on a printer:

```
enable printer-name
```

Only printers are enabled for printing (not classes). If you want to enable several printers at one time, list the printers, separated by spaces, on the same line as the `enable` command. Do not enclose the list in quotes.

At some point you may have to disable a printer. This should be done before you change the form or print wheel, or whenever you wish to stop what is currently printing. Disabling a printer stops further print requests from being printed, but it will not stop the LP print service from accepting new print requests for the printer. Normally, disabling a printer will also stop the request that is currently printing, placing it back in the queue so it can be printed later. However, you can have the LP print service wait until the current request finishes or even cancel the request outright.

Type one of the following commands to disable a printer:

```
disable -r "reason" printer-name
disable -W -r "reason" printer-name
disable -c -r "reason" printer-name
```

The first command disables the printer, stopping the currently printing request and saving it for printing later. The other commands also disable the printer, but the second has the LP print service wait for the current request to finish, while the third cancels the current request. The *reason* is stored and displayed whenever anyone asks the status of the printer. You can leave it (and the `-r`) out if you do not want to give a reason.

Several printers can be disabled at once by listing their names in the same line as the `disable` command.

9.8.1 Allowing Users to Enable and Disable a Printer

You may want to make the `enable` and `disable` commands available for use by other people. This is useful, for instance, if you have a small organization where anyone who spots a problem with the printer should disable it and fix the problem. This is not a good idea if you want to keep others from interfering with the proper operation of the print services.

If you want to allow others access to the `enable` and `disable` commands, you make use of a standard UNIX System feature called the "setuid bit." By having these commands owned by the user `lp` (they should be already) and by setting the setuid bit, anyone will be allowed to use the files. Clearing the bit again removes this privilege.

To allow everybody to use these commands, type the following two commands:

```
chown lp /usr/bin/enable /usr/bin/disable
chmod u+s /usr/bin/enable /usr/bin/disable
```

The first command makes the user `lp` the owner of the commands; this should be redundant, but it is safer to run the command anyway. The second command turns on the setuid bit.

To prevent others from using these commands, type:

```
chmod u-s /usr/bin/enable /usr/bin/disable
```

9.9 Examining a Printer Configuration

Once you have defined a printer configuration, you probably want to review it to see if it is correct. If, after examining the configuration, you find you have made a mistake, just re-enter the command that applies to the part that is wrong.

Use the `lpstat` command to examine both the configuration and the current status of a printer. A short form of this command gives just the status; you can use it to see if the printer exists and if it is busy, idle, or disabled. A long form of the command adds the complete configuration.

Type one of the following commands to examine a printer:

```
lpstat -p printer-name
lpstat -p printer-name -l
```

The second command is the long form. With either command you should see one of the following lines of output:

```
printer printer-name now printing request-id. enabled
since date.

printer printer-name is idle. enabled since date.

printer printer-name disabled since date.
        reason

printer printer-name waiting for auto-retry.
        reason
```

The `waiting for auto-retry` output shows that the LP print service failed in trying to use the printer (because of the *reason* shown) and that the print service will try again later.

With the long form of the command, you should also see the following output:

```
Form mounted: form-name
Content types: content-type-list
Printer type: printer-type
Description: comment
Connection: connection-info
Interface: path-name
On fault: alert-method
After fault: fault-recovery
Users allowed:
        user-list
Forms allowed:
        form-list
Banner required
Character sets:
        character-set-list
Default pitch: integer CPI, integer LPI
Default page size: scaled-decimal-number wide,
scaled-decimal-number long
Default port settings: stty-option-list
```

9.10 Trouble Shooting

If you are having difficulty getting your printer to work, here are a few suggestions for what to do.

9.10.1 No Output – Nothing Prints

The printer is sitting idle; nothing happens. First, check the documentation that came with the printer to see if there is a self-test feature you can invoke; make sure the printer is working before continuing.

9.10.1.1 Is the Printer Connected to the Computer? Check to make sure that the printer is connected to the printer. Refer to your printer's owners' manual for installation instructions.

9.10.1.2 Is the Printer Enabled? The printer must be "enabled" in two ways. First, the printer must be turned on and ready to receive data from the computer. Second, the LP print service must be ready to use the printer. Set up the printer as described in section 9.6, "Printer Management." If you receive error messages when doing this, follow the "fixes" suggested in the messages. When you have finished setting up the printer, type:

```
/usr/lib/accept printer-name
enable printer-name
```

where *printer-name* is the name you assigned to the printer for the LP print service. Now submit a sample file for printing:

```
lp -d printer-name -T printer-type file-name
```

If you did not give a printer type for the printer, leave out the `-T` *printer-type* option.

9.10.1.3 Is the Baud Rate Correct? If the baud rate (the rate at which the computer sends data to the printer) is not matched with the printer, sometimes nothing will print. See below.

9.10.2 Illegible Output

The printer tries printing, but it is not what you expected and certainly is not readable.

9.10.2.1 Is the Baud Rate Correct? Usually when the baud rate does not match with the printer, you will get some output but it will not look at all like what you submitted for printing. Random characters will appear with an unusual mix of special characters and unlikely spacing.

Read the documentation that came with the printer to find out its baud rate. It should probably be set at 9600 baud for optimum performance, but that does not matter for now. If it is not set to 9600 baud, you can have the LP print service use the correct baud rate (by default it uses 9600). If the printer is connected via a parallel port, the baud rate does not matter.

To set a different baud rate for the LP print service to use, type:

```
/usr/lib/lpadmin -p printer-name -o "stty=baud-rate"
```

Now submit a sample file for printing (explained earlier in this section).

9.10.2.2 Is the Parity Setting Correct? Some printers use a "parity bit" to ensure that the data they receive for printing has not been garbled in transmission. The parity bit can be encoded in a few different ways, and both the computer and the printer must agree on which to use. If they do not agree, some characters will not be printed or will have another character substituted. However, usually the output looks approximately correct, with the spacing of "words" typical for your document and many letters in their correct place.

Check the documentation for the printer to see what it expects. If your printer is directly connected to the computer with a fairly short wire (50 feet or so), it does not have to use the parity bit, but it does not matter for now. The LP print service will not expect to set the parity bit by default. You can change this, however, by typing one of the following:

```
/usr/lib/lpadmin -p printer-name -o "stty=oddp"
/usr/lib/lpadmin -p printer-name -o "stty=evenp"
/usr/lib/lpadmin -p printer-name -o "stty=-parity"
```

The first command sets odd parity generation, the second sets even parity. The last command sets the default, no parity. Select the command that matches the needs of your printer.

If you are also setting a baud rate other than 9600, combine the baud rate setting with the parity settings, as in the following sample command:

```
/usr/lib/lpadmin -p printer-name -o "stty='evenp 1200'"
```

Both double and single quotes are needed.

9.10.2.3 Tabs Set Correctly? If the printer does not expect to receive tab characters, the output may be there but all jammed up against the right margin. See below.

9.10.2.4 Correct Printer Type? See section 9.10.4, "Wrong Character Set or Font."

9.10.3 Legible Printing, But Wrong Spacing

The output is all there, it is readable, but is double-spaced, has no left margin, is run together, or zig-zags down the page. These problems can be fixed by adjusting the printer settings (if possible) or having the LP print service use matching settings.

9.10.3.1 Double-Spaced. Either the `-onlcr` or `-tabs` option is needed.

```
/usr/lib/lpadmin -p printer-name -o "stty=-onlcr"
/usr/lib/lpadmin -p printer-name -o "stty=-tabs"
```

9.10.3.2 No Left Margin/Runs Together/Jammed Up. You need the `-tabs` option.

```
/usr/lib/lpadmin -p printer-name -o "stty=-tabs"
```

9.10.3.3 Zig Zags Down the Page. The `onlcr` option is needed. This is set by default, but you may have cleared it accidentally.

```
/usr/lib/lpadmin -p printer-name -o "stty=onlcr"
```

9.10.3.4 A Combination of Problems. If several of these options must be combined to take care of multiple problems, include them in one list as in the sample command below. Include any baud rate or parity settings, too.

```
/usr/lib/lpadmin -p printer-name -o "stty='-onlcr -tabs 2400'"
```

Both double and single quotes are needed.

9.10.3.5 Correct Printer Type? See below.

9.10.4 Wrong Character Set or Font

If the wrong printer type was selected when you set up the printer with the LP print service, the wrong "control characters" can be sent to the printer. The results are unpredictable and may cause output to disappear or be illegible, making it look like a problem described above. A simpler problem is that it sets the wrong character set or font.

If you do not know what printer type to give, try the following to examine the available printer types. First, if you think the printer type has a certain name, try the following command:

```
TERM=printer-type tput longname
```

(This may not work on early versions of UNIX System V.) The output of this command appears on your terminal and is a short description of the printer identified by the *printer-type*. Try the names you think might be right until you find one that identifies your printer.

If you do not know what names to try, you can examine the `/usr/lib/terminfo` directory to see what names are available. *Note that there are probably many names in that directory.* Type the following command to examine the directory:

```
ls -R /usr/lib/terminfo/*
```

Pick names from the list that match one word or number identifying your printer. For example, the name `495` would identify the AT&T 495 Printer. Try each of the names in the other command above.

When you have the name of a printer type you think is correct, set it in the LP print service by typing the following command:

```
/usr/lib/lpadmin -p printer-name -T printer-type
```

9.10.5 Dial-Out Failures

The LP print service uses the Basic Networking utilities to handle dial out printers. If a dialing failure occurs and you are receiving printer fault alerts, the LP print service reports the same error reported by the Basic Networking software for similar problems. (If you have not arranged to receive fault alerts, by default they are mailed to the user `lp`.)

9.10.6 Idle Printers

There are several reasons why you may find a printer idle and enabled but with print requests still queued:

- *The print requests need to be filtered.*
  Slow filters run one at a time to avoid overloading the system. Until a print request has been filtered (if it needs slow filtering), it will not print. Use the following command to see if the first waiting request is being filtered:

  ```
  lpstat -o -l
  ```

- *The printer has a fault.*
  Automatic continuation of printing after a fault has been detected does not occur immediately. The LP print service waits about 5 minutes before trying again and keeps trying until a request prints successfully. You can force a retry immediately by enabling the printer:

  ```
  enable printer-name
  ```

- *A dial-out printer was busy or did not answer, or all dial-out ports are busy.*
  As with automatic continuation after a fault, the LP print service waits 5 minutes before trying to reach a dial-out printer again. If the dial-out printer cannot be reached for an hour or two (depending on the reason), the LP print service finally alerts to a

possible problem. You can force a retry immediately by enabling the printer:

```
enable printer-name
```

- *Lost "child process."*
  If the UNIX System process controlling the printer is killed (by the system during periods of extremely heavy load or by an administrator), the LP print service may not realize it for a few minutes. Disabling the printer and then re-enabling it forces the LP print service to check for the controlling process and restart one. Make sure the printer is really idle because disabling a printer stops it in the middle of printing a request. Though the request is not lost, it has to be reprinted in its entirety.

  ```
  disable printer-name
  enable printer-name
  ```

  If the process that is lost is one controlling a slow filter, do not try re-enabling the printer; instead, put the print request (the one at the head of the queue for the printer) on hold and then resume it, as shown below:

  ```
  lpstat -o -l
  lp -i request-id -H hold
  lp -i request-id -H resume
  ```

  Use the first command to list the requests queued.

9.11 Managing the Printing Load

Occasionally, you may need to stop accepting print requests for a printer or move print requests from one printer to another. There are various reasons why you might want to do this, such as:

- The printer needs periodic maintenance.
- The printer is broken.
- The printer has been removed.
- You have changed the configuration so that the printer is to be used differently.
- Too many large print requests are queued for one printer and should be spread around.

If you are going to make a big change in the way a printer is to be used, such as stopping its ability to handle a certain form, changing the print wheels available for it, or disallowing some people from using it, print requests that are currently queued for printing on it have to be moved or canceled. The LP print service attempts to find alternate printers but only if the user does not care which printer is to be used. Such requests are not automatically moved; if you do not move them first, the LP print service cancels them.

If you decide that a printer is to be taken out of service, its configuration is to be changed, or it is too heavily loaded, you may want to move print requests off it and reject additional requests for it for awhile. Use the `lpmove` and `reject` commands for this. If you do reject requests for a printer, you can later accept requests using the `accept` command.

9.11.1 Rejecting Requests for a Printer or Class

To stop accepting any new requests for a printer or class of printers, type the following command:

```
/usr/lib/reject -r "reason" printer-or-class-name
```

You can reject requests for several printers or classes in one command by listing their names on the same line, separating the names with spaces. The *reason* is displayed whenever anyone tries to print a file on the printer. You can drop it (and the `-r`) if you do not want to give a reason.

Although the `reject` command stops any new print requests from being accepted, it does not move or cancel any requests currently queued for the printer. These continue to print as long as the printer is enabled.

9.11.2 Accepting Requests for a Printer or Class

After the condition that led to denying requests has been corrected or changed, type the following command to start accepting new requests:

```
/usr/lib/accept printer-or-class-name
```

Again, you can accept requests for several printers or classes in one command by listing their names on the same line.

You will always have to use the `accept` command for a new printer or class after you have added it because the LP print service does not initially accept requests for new printers or classes.

9.11.3 Moving Requests to Another Printer

If you have to move requests from one printer or class to another, type one of the following commands:

```
/usr/lib/lpmove request-id printer-name
/usr/lib/lpmove printer-name_1 printer-name_2
```

You can give more than one request ID before the printer name in the first command.

The first command above moves the listed requests to the printer named. The latter command moves *all* requests currently queued for the first printer to the second printer. When the latter command is used, the LP print service also no longer accepts requests for the first printer (this has the same effect as the `reject` command).

9.11.4 Examples

Here are some examples of how you might use these three commands:

9.11.4.1 Example 1. You have decided it is time to change the ribbon on printer `lp1` and perform some preventive maintenance. You want to move all the requests for printer `lp1` to printer `lp2`. After the requests are moved, the LP print service will no longer accept requests for `lp1` (this is the same effect as a `reject lp1` command issued after the `lpmove` command).

```
/usr/lib/lpmove lp1 lp2
```

(At this point you may disable the printer and start working on it.)

9.11.4.2 Example 2. You have finished changing the ribbon and the other work on `lp1`, so now you want to bring it back into service.

```
/usr/lib/accept lp1
```

(At this point, if you had disabled the printer you should re-enable it. See section 9.8, "Enabling and Disabling a Printer.")

9.11.4.3 Example 3. You notice that someone has queued several large files for printing on the printer `laser1`. Meanwhile `laser2` is currently idle because no one had queued requests for it. You will move the two biggest requests, `laser1-23` and `laser1-46`, to `laser2` and reject any new requests for `laser1` for the time being.

```
/usr/lib/lpmove laser1-23 laser1-46 laser2
/usr/lib/reject \-r "too busy--will reopen later"
laser1
```

9.12 Managing Queue Priorities

The LP print service provides a simple priority mechanism that people can use to adjust the position of a print request in the queue. Each print request can be given a priority level by the person who submits it; this is a number from 0 to 39, with *smaller* numbers indicating *higher* levels of priority. Requests with higher priority (smaller numbers) are placed ahead of requests with lower priority (larger numbers).

In this way, if you decide that your print request is too low in priority, you can set a higher priority (lower value) when you submit the file for printing. If you decide that your print request is too high in priority, you can set a lower priority (higher value) when you submit the file for printing.

A priority scheme this simple would not work if there were no controls on how high one can set the priority. You can define the following characteristics of this scheme:

- Each user can be assigned a priority limit. One cannot submit a print request with a priority higher than his or her limit, although one can submit a request with a lower priority.
- A default priority limit can be assigned for the balance of users not assigned a personal limit.
- A default priority can be set. This is the priority given print requests to which the user does not assign a priority.

By setting the characteristics according to your needs, you can prevent lower priority printing tasks (such as regular printing by most staff members) from interfering with higher priority printing tasks (such as payroll check printing by the accounting staff).

You may find that you want a critical print request to print ahead of any others, perhaps even if it has to pre-empt the currently printing request. You can have the LP print service give immediate handling to a print request and can have it put another print request on hold. This lets the first print request print and delays that latter print request until you have it "resumed."

The `lpusers` command lets you assign both priority limits for users and priority defaults. In addition, you can use the:

```
lp -i request-id -H hold
```

and

```
lp -i request-id -H immediate
```

commands to put a request on hold or move it up for immediate printing, respectively. These commands are discussed in detail later in this section.

9.12.1 Setting Priority Limits

To set a user's priority limit, type the following command:

```
/usr/lib/lpusers -q priority-level -u user-name
```

You can set the limit for a group of users by listing their names after the `-u` option. Separate multiple names with a comma or space (enclose the list in quotes if you use a space, though). *priority-level* is a number from 0 to 39. As mentioned before, the lower the number, the higher the priority or in this case the priority limit.

If you want to set the priority limit for all other users, type the following:

```
/usr/lib/lpusers -q priority-level
```

This sets the default limit; the default applies to those people who have not been given a personal limit using the earlier `lpusers` command.

If you later decide that someone should have a different priority limit, just re-enter the first command above with a new limit. If you decide that someone with a personal limit should have just whatever the default limit is, type:

```
/usr/lib/lpusers -u user-name
```

Again, you can do this for more than one person at a time by giving a list of names. Using the `lpusers` command with just the `-u` option puts the users in the "default limit" category.

If you do not set a default limit, people without personal limits are limited to priorities in the range of 20 to 39.

9.12.2 Setting a Default Priority

You can set the default priority that should be assigned to those print requests submitted without a priority. Use the following command:

```
/usr/lib/lpusers -d priority-level
```

Do not confuse this default with the "default limit." This default is applied when a user does not give a priority; the "default limit" is applied if you have not assigned a limit for a user – it is used to limit the user from giving too high a priority.

☛ If the default priority is greater than the limit for a user, the limit is used instead.

If you do not set a default priority, the LP print service uses the default of 20.

9.12.3 Examining the Priority Limits and Defaults

You can examine all the settings you have assigned for priority limits and defaults by typing:

```
/usr/lib/lpusers -l
```

9.12.4 Moving a Request Around in the Queue

Once a user has submitted a print request, you can move it around in the queue to some degree:

- You can adjust the priority to any level regardless of the limit for the user.
- You can put it on hold and let other requests print ahead of it.
- You can put it at the head of the queue for immediate printing.

Use the `lp` user command to do each of these.

9.12.5 Changing the Priority for a Request

A print request that is still waiting to print can be reassigned a new priority. This repositions it in the queue to put it ahead of lower priority requests, behind any others at the same or higher priority. The priority limit assigned to the user (or the default priority limit) has no effect because you will override this limit as the administrator.

Type the following command to change the priority of a request:

```
lp -i requestid -q new-priority-level
```

You can change only one request at a time with this command.

If a request is already printing, you cannot change its priority.

9.12.6 Putting a Request on Hold

Any request that has not finished printing can be put on hold. You can stop its printing, if it currently is printing, and keep it from printing until you resume it. A user can also put his or her own request on hold and then resume it, but cannot resume a print request you have put on hold.

Type the following command to place a request on hold:

```
lp -i request-id -H hold
```

Type the following command to resume the request:

```
lp -i request-id -H resume
```

Once resumed, a request continues to move up the queue and eventually prints. If it had been printing when you held it, it is the next request to print. Normally it starts printing from the beginning, with page one, but you can have it start printing at a later page. Type the following command to resume the request at a different page:

```
lp -i request-id -H resume -P starting-page-
```

The final dash is needed to specify the starting page and all subsequent pages.

☛ The ability to print a subset of pages requires the presence of a filter that can handle this. The default filter used by the LP print service does not. An attempt to resume a request on a later page is rejected if an appropriate filter is not being used.

9.12.7 Moving a Request to the Head of the Queue

You can move a print request to the head of the queue so that it is the next one eligible for printing. If it must start printing immediately but another request is currently printing, you can hold the other request as described above.

Type the following command to move a print request to the head of the queue:

```
lp -i request-id -H immediate
```

Only you can move a request like this; regular users cannot use the `-H immediate` option.

If you set more than one request for immediate printing, they print in the reverse order set; that is, the request moved to the head of the queue most recently prints first.

9.13 Forms

If you do not use special forms for printing, you can skip this section. This section tells you how you can manage the use of preprinted forms with the LP print service. You will see how you can:

- Define a new form
- Remove a form
- Restrict user access to a form
- Arrange alerting to the need to mount a form
- Mount a form
- Examine a form

Before getting into the details, we discuss what a form means in the context of the LP print service.

9.13.1 What Is a Form?

A preprinted form is a paper image of a blank form that you can load into your printer. An application typically generates a file that, when printed on the blank form, fills out the form. Common examples of forms are:

- Blank checks
- Vouchers
- Receipts
- Labels
- Company letterhead
- Special paper stock

Typically, several copies of the blank form are loaded into the printer either as a tray of single sheets or as a box of fan-folded paper.

The LP print service helps you manage the use of preprinted forms but does not provide your application any help in filling out a form. This is your application's sole responsibility. The LP print service, however, keeps track of which print requests need special forms mounted and which forms are currently mounted, and can alert you to the need to mount a new form.

9.13.2 Defining a Form

The first thing you have to do to add a new form is define its characteristics. This is a short list that helps the LP print service remind you how to deal with the form and tells the LP print service how to initialize the printer to print properly on the form. You need to know the following about the form:

Page length	The length of the form or of each page in a multi-page form. This can be expressed as the number of lines, or the size in inches or centimeters.
Page width	The width of the form expressed in columns, inches, or centimeters.
Number of pages	The number of pages in a multi-page form. The LP print service uses this number with a filter (if available) to restrict the alignment pattern to be a single form long. (See the description of alignment patterns below.) If no filter is available to truncate the alignment pattern, the LP print service skips that step.
Line pitch	How close together separate lines are on the form. This is expressed in either lines per inch or lines per centimeter.
Character pitch	How close together separate characters are on the form. Similarly, this is expressed in either characters per inch or characters per centimeter.
Character set choice	The character set, print wheel, or font cartridge that should be used when this form is used. A user can choose a different character set for his or her own print request using this form, or you can insist that only one character set be used.
Ribbon color	If the form should always be printed using a certain color ribbon, then the LP print service can remind you which color to use when you mount the form.
Comment	Any comment you wish to make about the form. This comment is available for people to see so

they can understand what the form is, when it should be used, and so on.

Alignment pattern A sample file that the LP print service uses to fill one blank form. When mounting the form, you can examine this sample to see if the printing is lined up properly on the form. If it is not, you can adjust the printer to get it lined up.

☛ The LP print service does not try to mask sensitive information in an alignment pattern. If you do not want sensitive information printed on sample forms – very likely the case when you align checks, for instance – then you should mask the appropriate data. The LP print service keeps the alignment pattern stored in a safe place, where only you (i.e., the user `lp` and the superuser) can read it.

When you have gathered this information about the form, enter it as input to the `lpforms` command. You may want to first record this information in your own file to make it easier to edit the information as you enter it. You can then give the file as input instead. However you enter it, you should present the information as follows:

```
Page length: scaled-number
Page width: scaled-number
Number of pages: integer
Line pitch: scaled-number
Character pitch: scaled-number
Character set choice: character-set-name,mandatory
Ribbon color: ribbon-color
Comment: comment
Alignment pattern: alignment-pattern
```

Except for the `Alignment pattern`, which must be the last information given, the information can appear in any order (although the *comment* must follow the `Comment:` line). If the *comment* has to contain a line beginning with any of the key phrases (`Page length`, `Page width`, and so on), you should precede it with a "greater than" character (>) to hide the key phrase. This means, however, that any initial "greater than" character will be stripped from the comment when it is displayed.

Not all of the information has to be given. Missing information is assigned the following defaults:

Item	*Default*
Page length	66 lines
Page width	80 columns
Number of pages	1
Line pitch	6
Character pitch	10
Character set choice	Any
Ribbon color	Any
Comment	No default
Alignment pattern	No default

Use one of the following commands to define the form:

```
/usr/lib/lpforms -f form-name -F file-name
/usr/lib/lpforms -f form-name -
```

The first command gets the form definition from a file; the second command gets the form definition from you through the standard input. *form-name* can be anything you choose as long as it contains fourteen or fewer letters, digits, and underscores.

If you need to change a form, just re-enter one of the same commands. You need only give the changed information; information you leave out will stay the same.

9.13.3 Removing a Form

The LP print service has no fixed limit to the number of forms you can define. However, it is a good idea to remove forms no longer appropriate for two reasons. It helps to avoid confusing the users, who would otherwise see a long list of obsolete forms when they are trying to choose the correct form, and it avoids extra processing by the LP print service, which must occasionally look through all the forms to perform certain tasks.

Type the following command to remove a form:

```
/usr/lib/lpforms -f form-name -x
```

9.13.4 Restricting User Access

You can limit the use of a form to a subset of people on your computer. You may want to do this, for instance, with sensitive forms such as checks, which only people in the payroll department or accounts payable department can use.

The LP print service uses the list of users allowed or denied for a form to restrict use of the form. The LP print service refuses a user's request to print a file with a form he or she is not allowed to use.

The method of listing the users allowed or denied access to a form is similar to the method used to list users allowed or denied access to the `cron` and `at` facilities. Refer to *crontab*(1). Briefly, the rules are as follows:

- An allow list is a list of those users allowed to use the form. A deny list is a list of those users denied access to the form.
- If the allow list is not empty, the deny list is ignored. If the allow list is empty, the deny list is used. If both lists are empty, there are no restrictions on who can use the form.
- Putting `any` or `all` into the allow list allows everybody to use the form; putting `any` or `all` into the deny list denies everybody, except the user `lp` and the superuser.

You can add names of users to either list using one of the following commands:

```
/usr/lib/lpforms -f form-name -u allow:user-list
/usr/lib/lpforms -f form-name -u deny:user-list
```

The *user-list* is a list of names of users separated by a comma or space. If you use spaces to separate the names, enclose the entire list (including the `allow:` or `deny:` but not the `-u`) in quotes. The first command adds the names to the allow list and removes them from the deny list. The second command adds the names to the deny list and removes them from the allow list. Using `allow:all` allows everyone; using `deny:all` denies everyone.

If you do not add user names to the allow or deny lists, the LP print service assumes that everybody can use the form.

9.13.5 Alerting to Mount a Form

If you define more forms than printers, you will obviously not be able to print files on all the forms simultaneously. This means that some print requests may be held in the queue until you mount the forms they need. You could periodically monitor the number of print requests pending for a particular form, but the LP print service provides an easier way: You can ask to be alerted when the number of requests waiting for a form has exceeded some threshold.

You can choose one of several ways to receive an alert:

- You can receive an alert via electronic mail. Refer to *mail*(1).
- You can receive an alert written to whatever terminal on which you are logged in. Refer to *write*(1).
- You can receive an alert through a program of your choice.
- You can receive no alerts.

☛ If you elect to receive no alerts, you are responsible for checking to see if any print requests have not printed because the proper form is not mounted.

In addition to the method of alerting, you can also set the number of requests that must be queued before you are alerted, and you can arrange for repeated alerts every few minutes until the form is mounted. You can choose the rate of repeated alerts, or can choose to receive only one alert per form.

To arrange for alerting to the need to mount a form, type one of the following commands:

```
/usr/lib/lpforms -f form-name -A mail -Q integer
-W minutes
/usr/lib/lpforms -f form-name -A write -Q integer
-W minutes
/usr/lib/lpforms -f form-name -A 'command' -Q integer
-W minutes
/usr/lib/lpforms -f form-name -A none
```

The first two commands direct the LP print service to send you a mail message or write the message directly to your terminal, respectively, for each alert. The third command directs the LP print service to run *command* for each alert. The shell environment currently in effect when you enter the third command is saved and restored for the execution of *command*; this includes the environment variables, user and group IDs, and current directory. The fourth command above directs the LP print service not to send you an alert when the form needs to be mounted. *integer* is the number of requests that need to be waiting for the form, and *minutes* is the number of minutes between repeated alerts.

☛ If you want mail sent or a message written to another person when a printer fault occurs, you must use the third command listed. Use the options:

```
        -A 'mail user-name'
```

or

```
        -A 'write user-name'
```

Once you start receiving repeated alerts, you can direct the LP print service to stop sending you alerts for the current case only by typing the following command:

```
/usr/lib/lpforms -f form-name -A quiet
```

Once the form has been mounted and unmounted again, alerts start again if too many requests are waiting. Alerts also start again if the number of requests waiting falls below the `-Q` threshold and then rises up to the `-Q` threshold again, as when waiting requests are canceled or if the type of alerting is changed.

If *form-name* is `all` in any of the commands above, the alerting condition applies to all forms.

If you do not define an alert method for a form, you will not receive an alert for it. If you do define a method but do not give the `-W` option, you are alerted once for each occasion.

9.13.6 Mounting a Form

Refer section 9.6.19, "Mounting a Form or Print Wheel."

9.13.7 Examining a Form

You can examine a form definition once you have added it to the LP print service. There are two commands to use, depending on the information you want to examine. The `lpforms` command displays the definition of the form. The display is suitable as input again, so that you can save the output in a file for future reference. The `lpstat` command displays the current status of the form.

Type one of the following commands to examine a defined form:

```
/usr/lib/lpforms -f form-name -l
/usr/lib/lpforms -f form-name -l >file-name
lpstat -f form-name
lpstat -f form-name -l
```

The first two commands present the definition of the form; the second command captures this definition in a file, which can later be used to redefine the form if you inadvertently remove the form from the LP print service. The last two commands present the status of the form, with the second giving a long form of output similar to the output of `lpforms -l`. Their output looks similar to this:

```
Page length: scaled-number
Page width: scaled-number
Number of pages: integer
Line pitch: scaled-number
Character pitch: scaled-number
Character set choice: character-set,mandatory
Ribbon color: ribbon-color
Comment: comment
Alignment pattern: content-type content
```

The `Alignment pattern` is not shown if the `lpstat` command is used to protect the potentially sensitive content.

9.14 Filter Management

This section tells you how you can manage the use of filters with the LP print service. You will learn how to:

- Define a new filter
- Change a filter
- Remove a filter
- Examine a filter

Section 9.19, "Customizing the Print Service," describes how you can write a filter.

9.14.1 What Is a Filter?

A filter plays three related roles:

- It converts a user's file into a data stream that will print properly on a given printer.
- It handles the various modes of printing that people may request with the `-y` option to the `lp` command, such as two-sided printing, landscape printing, draft or letter quality printing, and so on.
- It detects printer faults and informs the LP print service so that the latter can alert you.

Not every filter performs all three roles. However, given the printer-specific nature of these three roles, the LP print service has been designed so that these roles are separated out so that you, a printer manufacturer, or another source can provide these filters without having to change the LP print service.

A default filter is provided with the LP print service to provide simple printer fault detection; it does not convert files or handle any of the special modes. This may be adequate for your needs.

9.14.2 Converting Files

The LP print service allows you to "type" each printer you add to the system and allows a user to type each file he or she submits for printing. This type information is used to match a file with the printer that will best reproduce the file. Since many applications can generate data for various printers, this is often sufficient. However, not all of the applications you will use may generate output that will work on your printers.

By defining and creating a filter that converts such output into a type that your printers can handle, you can begin to support more applications in the LP print service. The Terminal Filters utilities provide a small set of simple filters that convert output from applications like `nroff` (from the Text Processing Workbench utilities) to data streams that print properly on some printers.

Each filter that is added to the system is "typed" as well, with the input type it can accept and the output type it can produce. Now the LP print service can be more sophisticated in its attempt to match a user's file with a printer. If it cannot find a direct match, it consults the table of filters to find one that will convert the file's type into the printer's type. Below are some examples.

9.14.2.1 Example 1. The user Chris has run a spreadsheet program and generated a file copy of a spreadsheet. Chris now wants to print this file using the LP print service. You have only AT&T model 455 printers on your system. Fortunately, the spreadsheet application understands how to generate output for several printers, and Chris knew to ask it to generate the file for the AT&T 455. When Chris submits the file for printing, the LP print service will queue it for one of the printers; no filter is needed.

9.14.2.2 Example 2. The user Marty has run the `nroff` word processing program to produce a copy of a large document. The `nroff` program also understands how to generate output for several printers, but Marty forgot and had it generate the default output type (let's call it type `nroff35`) which will not reproduce well on the AT&T 455. However, you had foreseen this situation and added the `450` filter to the filter table, marked it as taking standard `nroff` output (i.e., `nroff35`), and marked it as producing output for the AT&T 455 (let's call it type `455`). Since you added the printer as a type `455`, the LP print service recognizes that it can use the `450` filter to convert Marty's output before printing it.

9.14.3 Handling Special Modes

Another important role that filters can provide is the handling of the various printing modes that may be encountered. Each filter you add to the filter table can be registered as handling several aspects of printing. These are listed below:

Input type
Output type
Printer type
Character pitch
Line pitch
Page length
Page width
Pages to print
Character set
Form name
Number of copies
Modes

A filter is not required to handle most of these, only the modes. The LP print service provides a default handling for all the rest. However, it may be more efficient to have a filter handle these, or it may be that a filter has to know several of these aspects if it is to fulfill its other roles properly. A filter may need to know, for example, the page size and the print spacing if it is going to break up the pages in a file to fit on the printed pages. As another example, some printers can handle multiple copies more efficiently than the LP print service can, so a filter that is controlling the printer can use the number of copies information to skip the LP print service's default handling of this.

9.14.4 Detecting Printer Faults

Just as converting a file and handling special printing modes is a printer-specific role, so is the detecting of printer faults. The LP print service attempts to do this in general, and for most printers it properly detects faults. However, it is limited to checking for "hang-ups" (loss of carrier or the signal that indicates the printer is on-line) and excessive delays in printing (i.e., receipt of an XOFF flow-control character to shut off the data flow with no matching XON to turn the flow back on). It also cannot determine the cause of the fault, so it cannot tell you what to look for.

A properly designed filter can provide better fault coverage. Some printers are able to send a message to the host describing the reason for a fault. Others indicate a fault by other than dropping carrier or shutting off data flow. A filter can serve you by giving more information about a fault and detecting more of them.

Another benefit a filter can give is to wait for a printer fault to clear and to resume printing. This allows for more efficient printing when a fault occurs because the print request that was interrupted does not have to be reprinted in its entirety. Only a real filter, which has knowledge of the control sequences used by a printer, can know where a file breaks into pages; thus, only the filter knows how far back to go in the file to restart properly.

The LP print service has a simple interface that lets the filter get the fault information to you and restart if it can. The alerting mechanism (see section 9.6.11, "Fault Alerting") is handled by the LP print service; the interface program that manages the filter takes all error messages from the filter and places them into an alert message that can be sent to you. Thus, you see any fault descriptions that the filter puts out. If you have set the printer configuration so that printing should automatically resume after a fault is cleared, the interface program keeps the filter active so that it can pick right up where it left off.

9.14.5 Will Any Program Make a Good Filter?

It is tempting to use a program like `troff`, `nroff`, or a similar word processing program as a filter. However, the `troff` and `nroff` programs have a feature that allows people to reference additional files, called "include files," in the source document. The LP Spooler does not know about these files and will not enqueue them with the source document. The `troff` or `nroff` program may fail because it cannot access these additional files. Other programs may have similar features that limit their use as filters.

Here are guidelines to help you choose a good filter:

1. Examine the kinds of files people will submit for printing that will have to be processed by the filter. If they stand alone, that is, if they do not reference other files that the filter will need, the filter is probably okay. Check also to see if the filter expects any other files except those submitted by a user for printing.

2. If there can be referenced files inside the files submitted for printing or if the filter needs files other than those submitted by a user, then the filter is likely to fail because it is not able to access the additional files. We suggest you do not use the program as a filter but have each user run the program before submitting the files for printing.

Referenced files that are always given with full path names *may* be all right but only if the filter is used for local print requests. When used on requests submitted from a remote machine for printing on your machine, the filter may still fail if the referenced files are only on the remote machine.

9.14.6 Defining a Filter

There are several aspects of a filter that you have to define for the LP print service. These are listed below:

Input types

This is the list of file types that the filter can process. Most filters can take only one input type, but the LP print service does not restrict them to one. Several file types may be similar enough for the filter that it can deal with them. You can use whatever names you like here, subject to a limit of 14 letters, digits, and dashes (no underscore). Because the LP print service uses these names to match a filter with a file type, you should be consistent in the naming convention. For example, if more than one filter can accept the same input type, use the same name.

Your users should be made aware of these names so they know how to name their file's type when they submit the file for printing.

Output types

This is the list of file types that the filter can produce as output. For each file, the filter produces a single output type, but it may be able to vary that type on demand. The names of the output types are also restricted to 14 letters, digits, and dashes.

These names should either match the types of printers you have on your system or should match the input types handled by other filters. The LP print service gangs filters together in a shell pipeline to produce a new filter if it finds that several passes by different filters are needed to convert a file. It is unlikely that you will need this level of sophistication, but the LP print service allows it. Try to find a set of filters that takes as input types all the different files your users may want printed and that converts those files directly into types your printers can handle.

Printer types

This is a list of printer types into which the filter can convert files. While for most filters this list is identical to the output types, it can be different.

For example, you may have a printer that is given a single type for purposes of initialization (refer to the printer information in section 9.6, "Printer Management") but which can recognize several different types of files. In essence, these printers have an internal filter that converts the various types into one with which they can deal. Thus, a filter may produce one of several output types that match the file types that the printer can handle. The filter should be marked as working with that printer type.

As another example, you may have two different models of printers that are listed as both accepting the same types of files. However, due to slight differences in the manufacturing, one printer deviates in the results it produces. You label the printers as being of different printer types, say A and B, where B is the one that deviates. You create a filter that adjusts files to account for the deviation produced by printers of type B. Since this filter is only needed for those printer types, you list it as working only on type B printers.

For most printers and filters, you can leave this part of the filter definition blank.

Printers

You may have some printers that, although they are of the correct type for a filter, are in other ways inadequate for the output that the filter produces. For instance, you may want to dedicate one printer for fast turnaround; only files that the printer can handle without filtering are to be sent to that printer. Other printers, of identical type, you allow to be used for files that may need extensive filtering before they can be printed. You label the filter as working with only the latter printers.

In most cases, the filter should be able to work with all printers that accept the output that the filter produces, so you can leave this part of the filter definition blank.

Filter type

The LP print service recognizes "fast" filters and "slow" filters. Fast filters are labeled fast either because they incur little overhead in preparing a file for printing or because they must have access to the printer when they run. A filter that is to detect printer faults must be a fast filter. Slow filters are filters that incur a lot of overhead in preparing a file and do not have to have access to the printer. The LP print service runs slow filters in the background without tying up a printer. This allows files that need at most fast filtering (or no filtering) to move ahead; printers are not left idle while a slow filter works on a file if other files can be printed.

Command

This is the full path name of the program (filter) to run. If there are any fixed options that the program will always need, you can include them here.

Options

Options that the filter program will need, depending on the various modes and other aspects of printing, can be registered with the filter. This is discussed in more detail below.

When you have gathered this information about the filter, enter it as input to the `lpfilter` command. You may want to first record this information in your own file to make it easier to edit the information as you enter it. You can then give the file as input instead. However you enter it, you should present the information in the following way:

```
Input types: input-type-list
Output types: output-type-list
Printer types: printer-type-list
Printers: printer-list
Filter type: fast or slow
Command: simple-command
Options: template-list
```

The information can appear in any order. Not all the information has to be given. Missing information is assigned the following defaults:

Item	*Default*
Input types	Any
Output types	Any
Printer types	Any
Printers	Any
Filter type	Slow
Command	No default
Options	None

As you can see, the defaults define a very flexible filter, so you probably have to supply at least the input and output type(s). When you enter a list, separate the items in the list with blanks or commas.

9.14.7 Templates

All the information has been explained except the *templates-list*. Here is how the modes and printing aspects are registered.

The *templates-list* is a list of templates separated by commas and with the following form:

keyword pattern = replacement

keyword must be one of those listed in the following table; it labels the template as registering a particular characteristic of the printing. *pattern* is either a value of the characteristic or an asterisk (*) that stands as a place-holder for any value.

Characteristic	*Keyword*	*Possible Patterns*
Content type (input)	`INPUT`	*content-type*
Content type (output)	`OUTPUT`	*content-type*
Printer type	`TERM`	*printer-type*
Character pitch	`CPI`	*integer*
Line pitch	`LPI`	*integer*
Page length	`LENGTH`	*integer*
Page width	`WIDTH`	*integer*
Pages to print	`PAGES`	*page-list*
Character set	`CHARSET`	*character-set*
Form name	`FORM`	*form-name*
Number of copies	`COPIES`	*integer*
Modes	`MODES`	*mode*

The source of the values for these templates are as follows:

- The values of the `INPUT` and `OUTPUT` templates come from the file type that needs to be converted by the filter and the output type that has to be produced, respectively. Each is a type registered with the filter.
- The value for the `TERM` template is the printer type.
- The values for the `CPI`, `LPI`, `LENGTH`, and `WIDTH` templates come from the user's request, the form being used, or the defaults for the printer.
- The value for the `PAGES` template is a list of pages that should be printed. Typically it is a list of page ranges, either a pair of numbers or a single number, each range separated by a comma (e.g., 1-5,6,8,10 for pages 1 through 5, 6, 8, and 10). However, whatever value was given in the `-P` option to a print request is passed unchanged.
- The value for the `CHARSET` template is the name of the character set to be used.
- The value for the `FORM` template is the name of the form being printed on, if any.
- The value of the `COPIES` template is the number of copies of the file that should be made. If the filter uses this template, the LP print service reduces the number of copies of the filtered file

it will print to 1 since this single copy will really be the multiple copies produced by the filter.

- The value of the MODES template comes from the -y option of the lp command, the command a person uses to submit a print request. Since a user can give several -y options, there may be several values for the MODES template. The values are applied in the left-to-right order given by the user.

The replacement shows how the value of a template should be given to the filter program. It is typically a literal option, sometimes with the placeholder * included to show where the value goes. A few examples show how this works.

9.14.7.1 Example 1. The filter program is called /usr/bin/npf. It takes two input types, nroff37 and X, produces an output type called TX, and will work with any printer of type TX. The program accepts three options:

-Xb only for the input type X

-l *integer* for the length of the output page

-w *integer* for the width of the output page

The filter definition would look like this:

```
Input types: X,nroff37
Output types: TX
Printer types: TX
Command: /usr/bin/npf
Options: INPUT X = -Xb, LENGTH * = -l*,
WIDTH * = -w*
```

If a user submits a file of type nroff37 and asks that it be printed by a printer named lp1, which is of type TX, and requests a page length of 72,

```
lp -T nroff37 -d lp1 -o length=72
```

then this filter is called by the LP print service to convert the file. The filter is invoked as:

```
/usr/bin/npf -l72
```

9.14.7.2 Example 2. Another user submits a file of type X that is to be printed on the same printer, with default length and width. The filter is invoked as:

```
/usr/bin/npf -Xb
```

9.14.7.3 Example 3. The filter program is called `/usr/bin/x9700`. It takes one input type, `troff`, produces an output type called `9700`, and will work with any printer of type `9700`. The program has one fixed option, `-ib`, and accepts three other options:

`-l` *integer* for the length of the output page

`-s` *name* for the character set

`-o portrait` for portrait orientation of the paper

`-o landscape` for landscape orientation of the paper

You have decided that your users need give just the abbreviations `port` and `land` when they ask for the paper orientation. Since these are not options intrinsic to the LP print service, users will specify them using the `-y` option to the `lp` command.

The filter definition looks like this:

```
Input types: troff
Output types: 9700
Printer types: 9700
Command: /usr/bin/x9700 -ib
Options: LENGTH * = -l *, CHARSET * = -s *,
         MODES port = -o portrait, MODES land
         = -o landscape
```

(The last line is split into three lines for readability in this document. It would be entered as a single line.)

If a user submits a file of type `troff` for printing on a printer of type `9700` and requests landscape orientation using the `gothic` character set,

```
lp -T troff -S gothic -y land
```

then this filter is invoked by the LP print service to convert the file as follows:

```
/usr/bin/x9700 -ib -s gothic -o landscape
```

☛ If a pattern or replacement must include a comma or equals sign (=), escape its special meaning by preceding it with a backslash. A backslash in front of these two characters will be removed when the pattern or replacement is used. (All other backslashes are left alone.)

9.14.8 Command to Enter

Once you have a filter definition complete, enter one of the following commands to add the filter to the system:

```
/usr/lib/lpfilter -f filter-name -F file-name
/usr/lib/lpfilter -f filter-name -
```

The first command gets the filter definition from a file, and the second command gets the filter definition from you through the standard input. *filter-name* can be anything you choose, as long as it contains 14 or fewer letters, digits, and underscores.

If you need to change a filter, just re-enter one of the same commands. You need only give the changed information; information you leave out stays the same.

9.15 Removing a Filter

The LP print service has no fixed limit to the number of filters you can define. However, it is a good idea to remove filters no longer applicable to avoid extra processing by the LP print service, which must examine all filters to find one that works in a given situation.

Enter the following command to remove a filter:

```
/usr/lib/lpfilter -f filter-name -x
```

9.16 Examining a Filter

You can examine a filter definition once you have added it to the LP print service. The `lpfilter` command displays the definition of the filter in a form suitable as input again so that you can save the output in a file for future reference.

Type one of the following commands to examine a defined filter:

```
/usr/lib/lpfilter -f filter-name -l
/usr/lib/lpfilter -f filter-name -l >file-name
```

The first command presents the definition of the filter on your screen; the second command captures this definition in a file, which can later be used to redefine the filter if you inadvertently remove the filter from the LP print service.

9.17 A Word of Caution

Adding, changing, or deleting filters can cause print requests still queued to be canceled. This is because the LP print service evaluates each print request still queued to see which are affected by the filter change. Requests that are no longer printable, because a filter has been removed or changed, are canceled (with notifications sent to the people who submitted them). There can also be a delay in the response to new or changed print requests when filters are changed, due to the many characteristics that must be evaluated for each print request still queued. This delay can become noticeable if there are a large number of requests needing filtering.

Because of this possible impact, you may want to make changes to filters during periods when the LP print service is not being used much.

9.18 Directories and Files

This section lists the directories and files used by the LP print service. You can use this list to see if any files are missing or if the ownership or access permissions have changed. Normal operation of the LP print service should not cause any problems. However, if you do notice any discrepancies, it could be a cause for a security breach on your system.

At the end of this section is a description of the script used to clean out the request log periodically. You may want to change this script to have the file cleaned out on a different period or to condense the information into a report. Refer to section 9.18.1, "Cleaning Out the Request Log."

All directories and files are found under the parent directory `/usr/spool/lp`. This directory should have the following access permissions and ownership:

Permissions	*Owner*	*Group*	*Directory or File*
`drwxrwxr-x`	`lp`	`bin`	`/usr/spool/lp`

You can check this by typing:

```
ls -ld /usr/spool/lp
```

Under this directory you should see only the directories and files shown in the following table. Those marked with an asterisk (*) may be missing, depending on the state of the print service or its configuration.

You can generate a similar table for comparison by typing:

```
ls -lR /usr/spool/lp
```

Permissions	*Owner*	*Group*	*Directory or File*
-rw-rw-r--	lp	bin	* SCHEDLOCK
drwxrwxr-x	lp	bin	admins
drwxrwxr-x	lp	bin	bin
-rw-r--r--	lp†	bin†	* default
drwxrwxr-x	lp	bin	fifos
drwxrwxr-x	lp	bin	logs
drwxrwxr-x	lp	bin	model
drwxrwxr-x	lp	bin	requests
drwxrwxr-x	lp	bin	system
drwxrwxr-x	lp	bin	temp
-rw-r--r--	lp†	bin†	* users
/usr/spool/lp/admins:			
drwxrwxr-x	lp	bin	lp
/usr/spool/lp/admins/lp:			
drwxrwxr-x	lp	bin	classes
-rw-rw-r--	lp†	bin†	* filter.table
-rw-rw-r--	lp	bin	* filter.table.i
drwxrwxr-x	lp	bin	forms
drwxrwxr-x	lp	bin	interfaces
drwxrwxr-x	lp	bin	logs
drwxrwxr-x	lp	bin	printers
drwxrwxr-x	lp	bin	pwheels
/usr/spool/lp/admins/lp/classes:			
-rw-rw-r--	lp†	bin†	* *class1*
-rw-rw-r--	lp†	bin†	* *class2*
.			
.			
.			
-rw-rw-r--	lp†	bin†	* *classN*
/usr/spool/lp/admins/lp/forms:			
drwxrwxr-x	lp†	bin†	* *form1*
drwxrwxr-x	lp†	bin†	* *form2*
.			
.			
.			
drwxrwxr-x	lp†	bin†	* *formN*
/usr/spool/lp/admins/lp/forms/*formK*:			
-rwxrwx---	lp†	bin†	* alert.sh
-rw-rw----	lp†	bin†	* alert.vars
-rw-rw----	lp†	bin†	* align_ptrn
-rw-rw-r--	lp†	bin†	* allow
-rw-rw-r--	lp†	bin†	* comment
-rw-rw-r--	lp†	bin†	* deny
-rw-rw-r--	lp†	bin†	* describe
/usr/spool/lp/admins/lp/interfaces:			
-rwxrwxr-x	lp†	bin†	* *printer1*

```
-rwxrwxr-x                                  lp†    bin†    * printer2
.
.
.
-rwxrwxr-x                                  lp†    bin†    * printerN

/usr/spool/lp/admins/lp/printers:
drwxrwxr-x                                  lp†    bin†    * printer1
drwxrwxr-x                                  lp†    bin†    * printer2
.
.
.
drwxrwxr-x                                  lp†    bin†    * printerN

/usr/spool/lp/admins/lp/printers/printerK:
-rwxrwx---                                  lp†    bin†    * alert.sh
-rw-rw----                                  lp†    bin†    * alert.vars
-rw-rw-r--                                  lp†    bin†    * comment
-rw-rw-r--                                  lp†    bin†    * configuration
-rw-rw-r--                                  lp†    bin†    * forms.allow
-rw-rw-r--                                  lp†    bin†    * forms.deny
-rw-rw-r--                                  lp†    bin†    * users.allow
-rw-rw-r--                                  lp†    bin†    * users.deny

/usr/spool/lp/admins/lp/pwheels:
drwxrwxr-x                                  lp†    bin†    * printwheel1
drwxrwxr-x                                  lp†    bin†    * printwheel2
.
.
.
drwxrwxr-x                                  lp†    bin†    * printwheelN

/usr/spool/lp/admins/lp/pwheels/printwheelK
-rwxrwx---                                  lp†    bin†    * alert.sh
-rw-rw----                                  lp†    bin†    * alert.vars

/usr/spool/lp/bin:
-r--r--r--                                  lp     bin     alert.proto
-rwxrwxr-x                                  lp     bin     drain.output
-rwxrwxr-x                                  lp     bin     lp.cat
-rwxrwxr-x                                  lp     bin     lp.page
-rwxrwxr-x                                  lp     bin     lp.set
-rwxrwxr-x                                  lp     bin     lp.tell
-rwxrwxr-x                                  lp     bin     lpsched.jr
-rwxrwxr-x                                  lp     bin     slow.filter

/usr/spool/lp/fifos:
p-w--w--w-                                  root   other   * FIFO
drwxrwx--x                                  lp     sys     private
drwxrwx-wx                                  lp     sys     public

/usr/spool/lp/fifos/private:
pr--------                                  user   group   * machPID
.
.
```

```
/usr/spool/lp/fifos/public:
pr--------                                  user    group   * machPID
.
.
.

/usr/spool/lp/logs:
-rw-rw----                                  lp      bin     * lpsched
-rw-rw----                                  lp      bin     * requests
-rw-rw----                                  lp      bin     * requests1
-rw-rw----                                  lp      bin     * requests2
.
.
.
-rw-rw----                                  lp      bin     * requestsN

/usr/spool/lp/model:
-rwxrwxr-x                                  bin     bin     1640
-rwxrwxr-x                                  bin     bin     5310
-rwxrwxr-x                                  bin     bin     dqp10
-rwxrwxr-x                                  bin     bin     dumb
-rwxrwxr-x                                  bin     bin     f450
-rwxrwxr-x                                  bin     bin     hp
-rwxrwxr-x                                  bin     bin     lqp40
-rwxrwxr-x                                  bin     bin     pprx
-rwxrwxr-x                                  bin     bin     prx
-rwxrwxr-x                                  bin     bin     standard

/usr/spool/lp/requests:
-rw-rw----                                  lp      bin     * id1-0
-rw-rw----                                  lp      bin     * id2-0
.
.
.
-rw-rw----                                  lp      bin     * idN-0

/usr/spool/lp/system:
-rw-rw-r--                                  lp      bin     * cstatus
-rw-rw-r--                                  lp      bin     * pstatus

/usr/spool/lp/temp:
-rw-------                                  lp      bin     * idN-0
-rw-------                                  lp      bin     * idN-1
-rw-------                                  lp      bin     * idN-2
.
.
.
-rw-------                                  lp      bin     * idN-M
-rw-------                                  lp      bin     * FidN-1
-rw-------                                  lp      bin     * FidN-2
.
.
.
-rw-------                                  lp      bin     * FidN-M
-rw-------                                  lp      bin     * idN
```

```
-rw-------                              lp      bin      * A-K
-rw-------                              lp      bin      * F-K
-rw-------                              lp      bin      * P-K
```

The italicized names, *printerN*, *formN*, *classN*, *printwheelN*, and *idN* are placeholders for a single printer, form, class, print wheel, and request ID, respectively. (*idN* is just the numeric part of the request ID.) There will be one set of these directories and files for each active printer, form, class, print wheel, and request on your system. The italicized letter *K* is a placeholder for an internal number; the `A-`*K*, `F-`*K*, and `P-`*K*, files are used to store alert messages.

The ownership and permissions of the *idN-M* request files under the `/usr/spool/lp/temp` directory will change during the life of a print request, alternating between the user who submitted the request and the `lp` ID.

The directories under the `/usr/spool/lp/fifos` directory contain named pipes used to communicate between the LP print service and commands such as `lpadmin`, `lpstat`, `lp`, and so on. These two directories must have the permission flags and ownership shown if the communication with the LP print service is to work. Every entry below these directories is given a unique name formed by combining the name of the system (the node name) and the process ID of the command. The uniqueness of the entry names prevents two or more people from accidentally sharing the same communications path.

9.18.1 Cleaning Out the Request Log

The directories `/usr/spool/lp/temp` and `/usr/spool/lp/requests` contain files that describe each request that has been submitted to the LP print service. Each request has two files, one in each directory, that contain information about the request. The information is split to put more sensitive information in the `/usr/spool/lp/requests` directory where it can be kept secure. The request file in the `/usr/spool/lp/temp` is safe from all except the user who submitted the request, while the file in `/usr/spool/lp/requests` is safe even from the submitting user.

These files remain in their directories only as long as the request is on the queue. Once the request is finished, the information in the files is combined and appended to the file /usr/spool/lp/logs/requests. This file is not removed by the LP print service but can be cleaned out periodically, using, for instance, the cron facility. See *crontab*(1).

The default crontab entry suggested with the LP print service system is shown below:

```
13 3 * * * cd /usr/spool/lp/logs; if [ -f
requests ]; then /bin/mv requests xyzzy; /bin/cp
xyzzy requests; >xyzzy; /usr/lbin/agefile -c2
requests; /bin/mv xyzzy requests; fi
```

(This is one line in crontab but is split into several lines here for readability.) What this entry does, briefly, is "age" the file, changing the name to *requests-1* and moving the previous day's copy to *requests-2*. The number 2 in the -c option to the agefile program will keep the log files from the previous two days, discarding older log files. By changing this number, you can change the amount of information saved. On the other hand, if you want the information saved more often or want to clean out the file more often than once a day, change the time when the crontab entry is run by changing the first two numbers. The current values, 13 and 3, cause the cleanup to occur at 3:13 AM each day.

The default crontab entry supplied is sufficient to keep the old print request records from accumulating in the spooling file system. You may want to condense information in the request log to produce a report on the use of the LP print service or to aid in generating accounting information. You can produce a different script that examines the file and extracts information just before the cleanup procedure.

The request log has a simple structure that makes it easy to extract data using common UNIX System shell commands. The requests are listed in the order in which they were printed and are separated by lines that give the request ID. Each line below the separator line is marked with a single letter that identifies the kind of information contained in the line. Each letter is separated from the data by a single space. See the following table for details.

Letter	*Content of Line*
=	This is the separator line, containing the request ID, the user and group IDs of the user, the total number of bytes in the original (unfiltered) files, and the time when the request was queued. These items are separated by commas and are in the order just named. The user ID, group ID, and sizes are preceded by the word `uid`, `gid`, or `size`, respectively.
C	The number of copies printed.
D	The printer or class destination or the word `any`.
F	The name of the file printed. This line is repeated for each file printed, and files are printed in the order given.
f	The name of the form used.
H	The type of special handling used, spelled out (`resume`, `hold`, `immediate`). The only useful value found in this line is `immediate`.
N	The type of alert used when the print request successfully completed. The type is the letter `M` if the user was notified by mail, or `W` if the user was notified by a message to his or her terminal.
O	The `-o` options.
P	The priority of the print request.
p	The list of pages printed.
r	This single letter line is present if the user asked for "raw" processing of the files (the `-r` option of the `lp` command.)
S	The character set or print wheel used.
s	The outcome of the request as a combination of individual bits expressed in hexadecimal form. While several bits are used internally by the spooler, the most important bits are:

0x0004	Slow filtering finished successfully.
0x0010	Printing finished successfully.
0x0040	The request was canceled.
0x0100	The request failed filtering or printing.

T	The title placed on the banner page.
t	The type of content found in the file(s).
U	The name of the user who submitted the print request.
x	The slow filter used for the request.
Y	The list of special modes to give to the filters used to print the request.
y	The fast filter used for the request.
z	The printer used for the request. This will differ from the destination (the D line) if the request was queued for any printer or a class of printers or if the request was moved to another destination by the LP print service administrator.

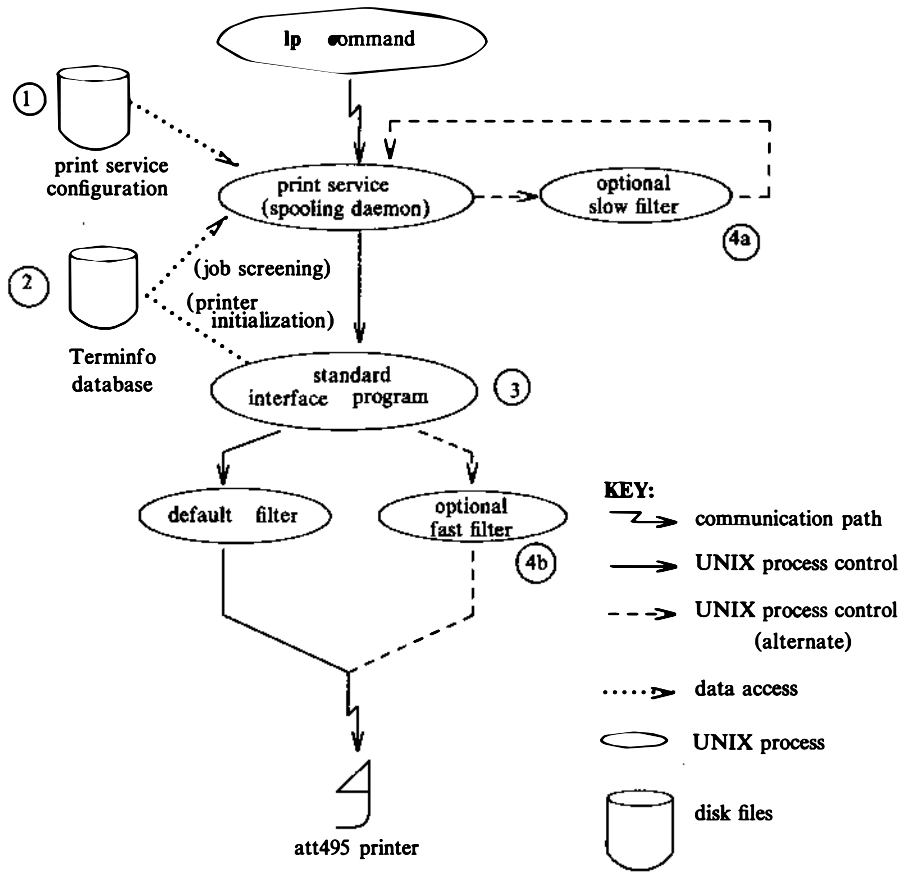
9.19 Customizing the Print Service

Although the LP print service tries to be flexible enough to handle most printers and printing needs, it cannot be complete. You may buy a printer that does not quite fit into the way the LP print service handles printers or may have a printing need that the standard features of the LP print service will not accommodate.

You can customize the LP print service in a few ways. This section tells you how you can:

- Adjust the printer port characteristics
- Adjust the Terminfo database
- Write an interface program
- Write a filter

The following figure gives an overview of the processing of a print request.



Each print request is sent to a spooling daemon that keeps track of all the requests. The daemon is created when you start the LP print service. This INTERACTIVE UNIX System process is also responsible for keeping track of the status of the printers and slow filters; when a printer finishes printing a user's file, the daemon starts it printing another request, if one is queued.

You are able to customize the print service by adjusting or replacing some of the pieces shown in the last figure. (The numbers are keyed to the figure.)

1. For most printers, you need only change the printer configuration stored on disk. Earlier sections of section 9 have explained how to do this. Some of the more printer-dependent

configuration data are the printer port characteristics: baud rate, parity, and so on.

2. For printers that are not represented in the Terminfo database, you can add a new entry that describes the capabilities of the printer. This database is used in two parallel capacities: screening print requests to ensure that those accepted can be handled by the desired printer and setting the printer in a state where it is ready to print the request.

   For instance, if the Terminfo database does not show a printer capable of setting a page length requested by a user, the spooling daemon will reject the request. On the other hand, if it does show that it is capable, then the same information is used by the interface program to initialize the printer.

3. For particularly difficult printers or if you want to add features not provided by the delivered LP print service, you can change the standard interface program. This program is responsible for managing the printer: it prints the banner page, initializes the printer, and invokes a filter to send copies of the user's files to the printer.

4a. and 4b.
To provide a link between the applications used on your system and the printers, you can add slow and fast filters. Each type of filter can convert a file into another form, mapping one set of escape sequences into another, for instance, and can provide special setup by interpreting print modes requested by a user. Slow filters are run separately by the daemon to avoid tying up a printer. Fast filters are run so their output goes directly to the printer; thus, they can exert control over the printer.

9.19.1 Adjusting the Printer Port Characteristics

You should make sure that the printer port characteristics set by the LP print service match the printer communication settings. The standard printer port settings were designed to work with typical UNIX System files and many printers, but they will not work with all files and printers. This is not really a customizing step, since a standard feature of the LP print service is to allow you to specify the port settings for each printer. However, it is an important step in getting your printer to work with the LP print service, so it is described in more detail here.

When you add a new printer, read the documentation that comes with it so that you understand what it expects from the host (the LP print service). Then read *stty*(1), which summarizes the various characteristics that can be set on a terminal or printer port.

Only some of the characteristics listed in *stty*(1) are important for printers. The ones likely to be of interest to you are listed below (but you should still consult the *stty*(1) for others).

`stty` *Option*	*Meaning*
`evenp`	Sends even parity in the 8th bit
`oddp`	Sends odd parity in the 8th bit
`-parity`	Does not generate parity, sends all 8 bits unchanged
`110 - 38400`	Sets the communications speed to this baud rate
`ixon`	Enables XON/XOFF (also known as START/STOP or DC1/DC3) flow control
`-ixon`	Turns off XON/XOFF flow control
`-opost`	Does not do any "output post-processing"
`opost`	Does "output post-processing" according to the settings listed below
`onlcr`	Sends a carriage return before every linefeed
`-onlcr`	Does not send a carriage return before every linefeed
`ocrnl`	Changes carriage returns into linefeeds
`-ocrnl`	Does not change carriage returns into linefeeds
`-tabs`	Changes tabs into an equivalent number of spaces
`tabs`	Does not change tabs into spaces

When you have a set of printer port characteristics you think should apply, adjust the printer configuration as described in section 9.6.7, "Printer Port Characteristics." You may find that the default settings are sufficient for your printer.

9.19.2 Adjusting the Terminfo Database

The LP print service relies on a standard interface and the Terminfo database to initialize each printer and set up a selected page size, character pitch, line pitch, and character set. Thus, it is usually sufficient to have the correct entry in the Terminfo database to add a new printer to the LP print service. Several entries for AT&T printers and other popular printers are delivered in Terminfo database entries with the LP print service package.

Each printer is identified in the Terminfo database with a short name; this kind of name is identical to the kind of name used to set the `TERM` shell variable. For instance, the AT&T model 455 printer is identified by the name `455`. The section titled "Acceptable Terminal Names" in Appendix F of the *User's Guide* gives a description of how to determine a correct `TERM` variable for a user terminal and can be used as a guide for picking a known name for your printer.

If you cannot find a Terminfo entry for your printer, you should add one. If you do not, you may still be able to use the printer with the LP print service, but you will not be able to get automatic selection of page size, pitch, and character sets, and you may have trouble keeping the printer set in the correct modes for each print request. Another option to follow instead of updating the Terminfo entry is to customize the interface program used with the printer. See the next section for details on how to do this.

There are hundreds of items that can be defined for each terminal or printer in the Terminfo database. However, the LP print service uses less than fifty of these, and most printers need even less than that. The following table lists the items that need to be defined (as appropriate for the printer) to add a new printer to the LP print service.

Terminfo Item	*Meaning*
Booleans:	
`daisy`	Printer needs operator to change character set
Numbers:	
`bufsz`	Number of bytes buffered before printing
`* cols`	Number of columns in a line
`* it`	Tabs initially every this many spaces
`* lines`	Number of lines on a page
`orc`	Horizontal resolution in units per character
`orhi`	Horizontal resolution in units per inch
`orl`	Vertical resolution in units per line
`orvi`	Vertical resolution in units per inch
`cps`	Average print rate in characters per second
Strings:	
`* cr`	Carriage return
`cpi`	Change number of characters per inch
`lpi`	Change number of lines per inch
`chr`	Change horizontal resolution
`cvr`	Change vertical resolution
`csnm`	List of character set names
`mgc`	Clear all margins (top, bottom, and sides)
`* hpa`	Horizontal position absolute
`* cud1`	Down one line
`* cuf1`	Carriage right
`swidm`	Enable double wide printing
`rwidm`	Disable double wide printing
`* ff`	Page eject
`* is1`	Printer initialization string
`* is2`	Printer initialization string
`* is3`	Printer initialization string
`* if`	Name of initialization file
`* iprog`	Path name of initializing program
`* cud`	Move carriage down # lines

Terminfo Item	*Meaning*
Strings:	
* `cuf`	Move carriage right # columns
* `rep`	Repeat a character # times
* `vpa`	Vertical position absolute
`scs`	Select character set
`smgb`	Set bottom margin at current line
`smgbp`	Set bottom margin
* `smgl`	Set left margin at current column
`smglp`	Set left margin
* `smgr`	Set right margin at current column
`smgrp`	Set right margin
`smgt`	Set top margin at current line
`smgtp`	Set top margin
`scsd`	Start definition of a character set
* `ht`	Tab to next 8 space tab stop

The items marked with a leading asterisk (*) are available on all releases of UNIX System V. The rest can be added only if you are using UNIX System V Release 3.2 or later.

☛ If you are running the LP print service on UNIX System V Release 3.1, only the Terminfo items marked in the table are available. They are sufficient for initializing the printer but not for setting page sizes and pitches or selecting character sets.

Consult *terminfo*(4) in the *INTERACTIVE SDS Guide and Programmer's Reference Manual* for its file structure and for details on how to construct a Terminfo database entry for a new printer.

Once you have made the new entry, you need to compile it into the database using the `tic` program. Just type the following command:

```
tic filename
```

filename is the name of the file containing the Terminfo entry you have crafted for the new printer. (This program is available in the Terminal Utilities.)

☛ The LP print service gains much efficiency by "caching" information from the Terminfo database. If you add or delete Terminfo entries or change the values that govern pitch settings, page width and length, or character sets, you should stop and restart the LP print service so it can read the new information.

9.19.3 How to Write an Interface Program

☛ If you have an interface program that you have used with the LP Spooler utilities before UNIX System V Release 3.2, it should still work with the LP print service. Note, however, that several `-o` options have been standardized and will be passed to every interface program. These may interfere with similarly named options your interface program uses.

If you have a printer that is not supported by simply adding an entry to the Terminfo database or if you have printing needs that are not supported by the standard interface program, you can furnish your own interface program. It is a good idea to start with the standard interface program and change it to fit, rather than starting from scratch. You can find a copy of it named `/usr/spool/lp/model/standard`.

9.19.3.1 What Does an Interface Program Do? Any interface program performs the following tasks:

- Initializes the printer port, if needed. The generic interface program uses the `stty` command to do this.
- Initializes the physical printer. The generic interface program uses the Terminfo and the `TERM` shell variable to get the control sequences to do this.
- Prints a banner page, if needed.
- Prints the correct number of copies of the request content.

An interface program is not responsible for opening the printer port. This is done by the LP print service, which calls a “dial-up” printer if that is how the printer is connected. The printer port connection is given to the interface program as standard output, and the printer is set to be the “controlling terminal” for the interface program so that a “hang-up” of the port causes a `SIGHUP` signal to be sent to the interface program.

A customized interface program must not terminate the connection to the printer or in any fashion “uninitialize” the printer. This allows the LP print service to use the interface program only for preparing the printer and printer port, while the printing of content is done elsewhere, by the LP print service, for example, for pre-printed form alignment patterns.

9.19.3.2 How Is an Interface Program Used? When the LP print service routes an output request to a printer, the interface program for the printer is invoked as follows:

```
/usr/spool/lp/admins/lp/interface/P id user title copies
options file1 file2 ...
```

(The last line is split into two lines for readability in this document.)

Arguments for the interface program are:

P	Printer name
id	Request ID returned by `lp`
user	Login name of user who made the request
title	Optional title specified by the user
copies	Number of copies requested by user
options	List of options separated by blanks, specified by user or set by the LP print service
file	Full path name of a file to be printed

When the interface program is invoked, its standard input comes from `/dev/null`, its standard output is directed to the printer port, and its standard error output is directed to a file that is given to the user who submitted the print request.

The standard interface recognizes the following values in the list in *options*:

- `nobanner`
  This option is used to skip the printing of a banner page; without it, a banner page is printed.
- `nofilebreak`
  This option is used to skip page breaks between separate data files; without it, a page break is made between each file in the content of a print request.
- `cpi=`*decimal-number*$_1$
- `lpi=`*decimal-number*$_2$ 1
  These options say to print with *decimal-number*$_1$ columns per inch and *decimal-number*$_2$ lines per inch, respectively. The

standard interface program extracts from the Terminfo database the control sequences needed to initialize the printer to handle the character and line pitches.

The words `pica`, `elite`, and `compressed` are acceptable replacements for the *decimal-number*$_1$ and are synonyms for *10* columns per inch, *12* columns per inch, and as many columns per inch as possible.

- `length=`*decimal-number*$_1$
- `width=`*decimal-number*$_2$ 1
  These options specify the length and width, respectively, of the pages to be printed. The standard interface program extracts from the Terminfo database the control sequences needed to initialize the printer to handle the page length and page width.
- `stty=`*'stty-option-list'*
  The *stty-option-list* is applied after a default *stty-option-list* as arguments to the `stty` command. The default list is used to establish a default port configuration; the additional list given to the interface program is used to change the configuration as needed.

The above options are either specified by the user when issuing a print request or by the LP print service from defaults given by the administrator for the printer (`cpi`, `lpi`, `length`, `width`, `stty`) or for the preprinted form used in the request (`cpi`, `lpi`, `length`, `width`).

Additional printer configuration information is passed to the interface program in shell variables:

- `TERM=`*printer-type*
  This shell variable specifies the type of printer. The value is used as a key for getting printer capability information from the extended Terminfo database.
- `FILTER=`*'pipeline'*
  This shell variable specifies the filter to use to send the request content to the printer; the filter is given control of the printer.
- `CHARSET=`*character-set*
  This shell variable specifies the character set to be used when printing the content of a print request. The standard interface program extracts from the Terminfo database the control sequences needed to select the character set.

A customized interface program should either ignore these options and shell variables or should recognize them and treat them in a consistent manner.

9.19.3.3 Customizing the Interface Program. You want to make sure that the custom interface program sets the proper `stty` modes (terminal characteristics such as baud rate or output options). The standard interface program does this, and you can follow suit. Look for the section that begins with the shell comment:

```
## Initialize the printer port
```

Follow the code used in the standard interface program. It sets both the default modes and the adjusted modes given by the LP print service or the user with a line similar to:

```
stty mode options 0<&1
```

This command line takes the standard input for the `stty` command from the printer port. An example of an `stty` command line that sets the baud rate at 1200 and sets some of the option modes is shown below:

```
stty -parenb -parodd 1200 cs8 cread clocal ixon 0<&1
```

One printer port characteristic not set by the standard interface program is hardware flow control. The way that this is set will vary depending on your computer hardware. The code for the standard interface program suggests where this and other printer port characteristics can be set. Look for the section that begins with the shell comment:

```
# Here you may want to add other port initialization code.
```

Because different printers have different numbers of columns, make sure the header and trailer for your interface program correspond to your printer. The standard interface program prints a banner that fits on an 80-column page (except for the user's title, which may be longer). Look for the section in the code for the standard interface program that begins with the shell comment:

```
## Print the banner page
```

The custom interface program should print all user-related error messages on the standard output or on the standard error. The messages sent to the standard error will be mailed to the user; the messages printed on the standard output end up on the printed page where they can be read by the user when he or she picks up the output.

When printing is complete, your interface program should exit with a code that tells the status of the print job. Exit codes are interpreted by the LP print service as follows:

Code	*Meaning to the LP Print Service*
0	The print request has completed successfully. If a printer fault has occurred, it has been cleared.
1 to 127	A problem was encountered in printing this particular request (for example, too many nonprintable characters or the request exceeds the printer capabilities). This problem does not affect future print requests. The LP print service notifies the person who submitted the request that there was an error in printing it. If a printer fault has occurred, it has been cleared.
128	Reserved for internal use by the LP print service. Interface programs must not exit with this code.
129	A printer fault was encountered in printing the request. This problem does affect future print requests. If the fault recovery for the printer directs the LP print service to wait for the administrator to fix the problem, it disables the printer. If the fault recovery is to continue printing, the LP print service does not disable the printer but tries printing again in a few minutes.
greater than 129	These codes are reserved for internal use by the LP print service. Interface programs must not exit with codes in this range.

As the table shows, one way of alerting the administrator to a printer fault is to exit with a code of 129. Unfortunately, if the interface program exits, the LP print service has no choice but to reprint the request from the beginning when the fault has been cleared. Another way of getting an alert to the administrator, but without requiring reprinting the entire request, is to have the interface program send a fault message to the LP print service but wait for the fault to clear. When the fault clears, the interface program can resume printing the user's file. When done printing, it can give a zero exit code just as if the fault never occurred. An added advantage is that the interface program can detect when the fault is cleared automatically so that the administrator does not have to enable the printer.

Fault messages can be sent to the LP print service using the `lp.tell` program. This is referenced using the `$LPTELL` shell variable in the standard interface code. The program takes its standard input and sends it to the LP print service where it is put into the message that alerts the administrator to the printer fault. If its standard input is empty, `lp.tell` does not initiate an alert. Examine the standard interface code immediately after these comments for an example of how the `lp.tell` (`$LPTELL`) program is used:

```
# Here's where we set up the $LPTELL program
to capture
# fault messages.

# Here's where we print the file.
```

With the special exit code 129 or the `lp.tell` program, there is no longer the need for the interface program to disable the printer itself. Your interface program can disable the printer directly, but doing so overrides the fault alerting mechanism. Alerts are sent only if the LP print service detects that the printer has faulted and the special exit code and the `lp.tell` program are its main detection tools.

If the LP print service has to interrupt the printing of a file at any time, it will "kill" the interface program with a signal 15 (see *kill*(1) and *signal*(2) in the *INTERACTIVE SDS Guide and Programmer's Reference Manual*). If the interface program dies from receipt of any other signal, the LP print service assumes that future print requests will not be affected and will continue to use the printer. The LP print service notifies the person who submitted the request that it did not finish successfully.

The signals `SIGHUP`, `SIGINT`, `SIGQUIT`, and `SIGPIPE` (trap numbers 1, 2, 3, and 13) start out being ignored when the interface is invoked. The standard interface changes this to trap these signals at appropriate times. The standard interface considers receipt of these signals as meaning the printer has a problem and issues a fault.

9.19.4 How to Write a Filter

A filter is used by the LP print service each time it has to print a type of file that is not acceptable by a printer. While a filter can be as simple or as complex as needed, there are only a few external requirements:

- The filter should get the content of a user's file from its standard input and send the converted file to the standard output.
- A slow filter can send messages about errors in the file to standard error; a fast filter should not, as described below. Error messages from a slow filter are collected and sent to the user who submitted the file for printing.
- If a slow filter dies because of receiving a signal, the print request is finished and the user who submitted the request is notified. Likewise, if a slow filter exits with a nonzero exit code, the print request is finished and the user is notified. The exit codes from fast filters are treated differently, as described below.
- A filter should not depend on other files that are not normally be accessible to a regular user; if the filter would fail if the user ran it directly, it will fail when the LP print service runs it.

Section 9.14, "Filter Management," describes how to add a filter to the LP print service.

There are a few more requirements if the filter is also to detect printer faults:

- If it can, it should wait for a fault to clear before exiting. In addition, it should continue printing at the top of the page where printing stopped after the fault clears. If this is not the administrator's intention, the LP print service stops the filter before alerting the administrator.
- It should send printer fault messages to its standard error as soon as the fault is recognized. It does not have to exit but can wait as described above.
- It should not send messages about errors in the file to standard error. Any messages on the standard error will eventually generate a pointer fault. These should be included in the standard output stream, where they can be read by the user.
- It should exit with a zero exit code if the user's file is finished (even if errors in the file prevented it from printing correctly).
- It should exit with a nonzero exit code only if a printer fault kept it from finishing a file.
- When added to the filter table, it must be added as a fast filter. Refer to section 9.14.6, "Defining a Filter."

10. ADDING MODEMS, PRINTERS, AND OTHER SERIAL DEVICES

10.1 Basic Networking Procedures

Adding Basic Networking involves the following:

1. Choose to physically connect your computer to another computer terminal, or other remote device using one of the following:

a direct link	Physically connect a null modem cable from a serial port on your computer to a port on another computer. See section 10.2.1, "Physical Connection of Computer to DTE Direct Link," for details.
a modem	Install and configure your modem as outlined in section 10.4.5, "Recommended Modem Settings," and section 10.2.3, "Physical Connection of Computer to Modem (DCE)," for details.

2. Establish the logical connection between the INTERACTIVE UNIX Operating System and your modem or direct link. This involves updating the appropriate support files to reflect the presence of a direct link or modem. Refer to section 10.2.2, "Basic Networking Software and Direct Links," and section 11, "BASIC NETWORKING ADMINISTRATION," for details.

10.2 Direct Links and Modems

This section discusses the following configurations:

- Computer to Data Terminal Equipment (DTE) direct link, such as another computer or terminal.
- A computer to Data Communications Equipment (DCE), such as a modem.

Your computer will connect to any other machine with an RS-232 port. Your computer will support any kind of auto dial modem.

An advantage of using a direct link is that the link is always available and the time required to access the link is short. Direct links are beneficial when:

- The two machines transfer large amounts of data on a regular basis.

- The two machines are located no more than several hundred cable feet apart.

The amount of cable used to link two machines depends on the environment in which the cable is run. The standard for RS-232 connections is 50 feet or less. As the cable length is increased, noise on the lines may become a problem. This means that the transmission rate must be decreased or limited distance modems should be placed on each end of the line. Generally, you should not use more than 1000 cable feet to connect the two machines, and even this usually requires special cabling and line signal boosters.

The advantage of using a modem is that a port is not dedicated to only one computer. You can also be networked to a remote computer located anywhere in the world where the telephone network exists. The disadvantages are that the port of the remote computer may be busy, the transmission rate is slower, and line quality is not guaranteed.

The following table shows a subset of the RS-232C standard pins for a DB25 connector, their names, and whether they function as input or output pins for a particular device (DTE or DCE). In the table, "in" means the signal on the line is generated by an external source and treated as input by the device; "out" means the device generates the signal on the pin. Note that 9-pin DB9 connectors have a different pin assignment. Logical pin functions, however, are identical.

Pin	*Description*	*Name*	*DTE*	*DCE*
1	Frame Ground	–	–	–
2	Transmitted Data	TD	out	in
3	Received Data	RD	in	out
4	Request to Send	RTS	out	in
5	Clear to Send	CTS	in	out
6	Data Set Ready	DSR	in	out
7	Signal Ground	–	–	–
8	Data Carrier Detect	DCD	in	out
9	Data Terminal Ready	DTR	out	in

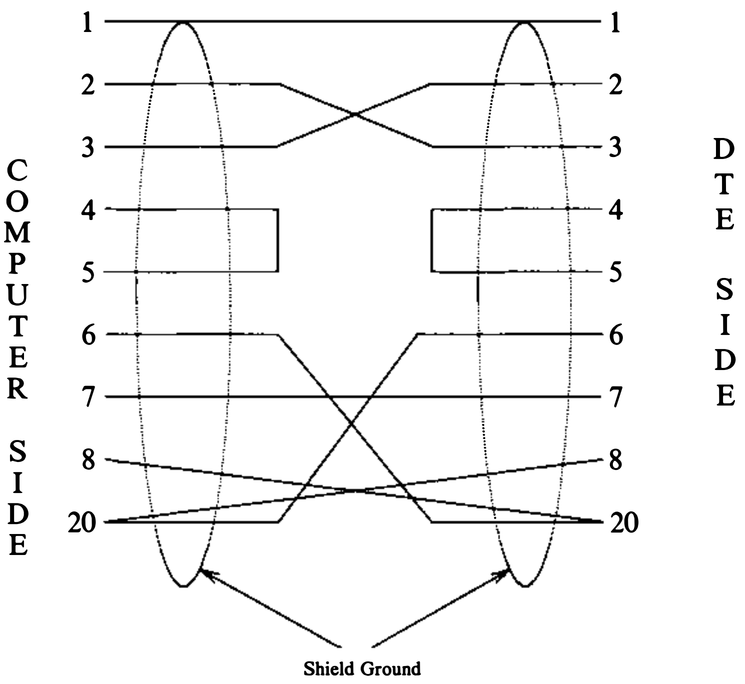
10.2.1 Physical Connection of Computer to DTE Direct Link

Connecting a computer to another RS-232C device (DTE) (for example, another computer) requires the use of a null-modem cable that must be constructed as follows:

Pin 1 to 1
Pin 2 to 3
Pin 3 to 2
Strap pin 4 to 5 in the same plug
Pin 6 to 20
Pin 7 to 7
Pin 8 to 20
Pin 20 to 6
Pin 20 to 8.

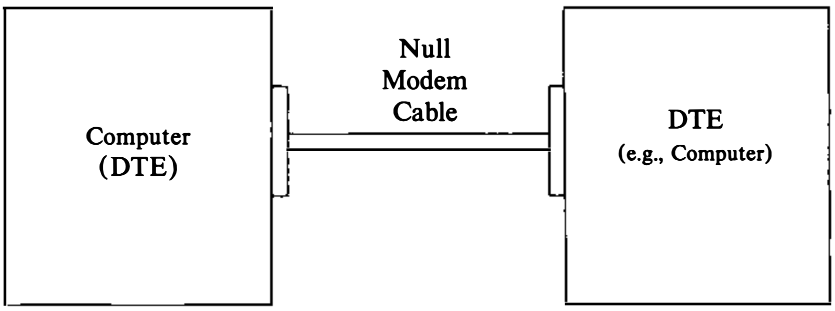
In a null-modem cable, input pins are connected to output pins to simulate the existence of an intermediate modem pair in the connection.

10.2.1.1 Wiring for Direct Link. Null-modem cables are commercially available for the direct link. If you desire to customize your own null-modem cable, nine leads must be wired for a connection to be made as shown in the following figure. Do not attach wiring to unused signals.



DTE--Data Terminal Equipment

The next figure shows a simple illustration of how a computer connects to a DTE direct link.



DTE -- Data Terminal Equipment

10.2.2 Basic Networking Software and Direct Links

This section describes the software files that must be modified on your computer in order to accommodate a direct link connection. Consult the documentation provided with your machine if you are linking directly to a remote machine other than a computer.

The following support files must be updated to reflect the presence of a direct link:

- `/usr/lib/uucp/Devices`
- `/etc/inittab`
- `/usr/lib/uucp/Systems`

10.2.3 Physical Connection of Computer to Modem (DCE)

A DCE device such as a modem can connect to your computer with an RS-232 cable. The computer's serial connector usually has a DTE configuration and the modem is required to have a DCE configuration. The pin connections for modem cable are shown in the table in section 10.2.1. The pin connections for the RS-232 modem cable are:

Pin 1 to 1
Pin 2 to 2
Pin 3 to 3
Pin 6 to 6
Pin 7 to 7
Pin 8 to 8
Pin 20 to 20

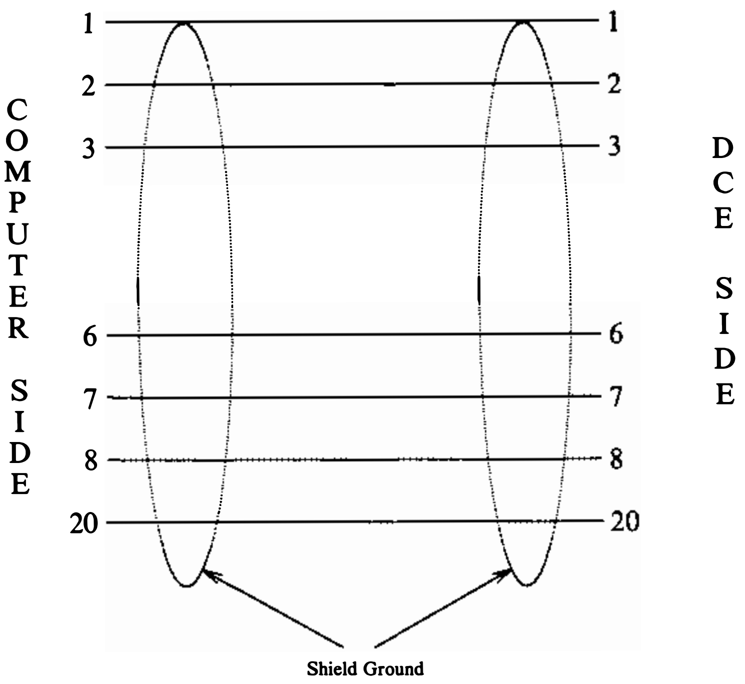
DB-9 (DTE) to DB-25 (DCE)		
DB9 Pin (DTE)	*Signal Name*	*DB25 Pin (DCE)*
1	Carrier Detect (DCD)	8
2	Received Data (RD)	3
3	Transmitted Data (TD)	2
4	Data Terminal Ready (DTR)	20
5	Signal Ground	7
6	Data Set Ready (DSR)	6
7	Request to Send (RTS)	4
8	Clear to Send (CTS)	5
9	Ring Indicator (RI)	22 (not used)

Note that the following signals must be connected for a direct line: TD, RD, ground; and for a modem: TD, RD, ground, CD, DTR.

DB-25 (DTE) to DB-25 (DCE)		
DB25 Pin (DTE)	*Signal Name*	*DB25 Pin (DCE)*
1	–	Chassis Ground
2 (out)	TD	2
3 (in)	RD	3
4 (out)	RTS	4
5 (in)	CTS	5
6 (in)	DSR	6
7	Signal Ground	7
8 (in)	DCD	8
20 (out)	DTR	20
22 (in)	RI	22

Note that this cable is wired straight through, so a ribbon cable and a pair of Insulation Displacement Connectors (IDC) can be used to make an instant cable. Ribbon cable is not shielded, though, so be sure to keep the length short – a few feet at most.

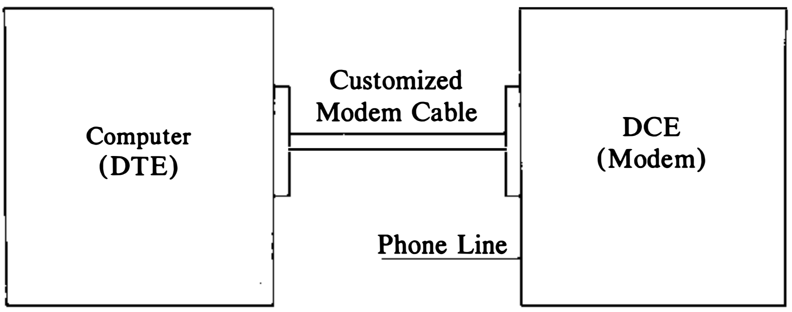
10.2.3.1 Wiring for Modems. Wire to pins only used at both ends. Do not attach wiring to an unused signal. Seven leads must be wired to customize modem cable as shown in the following figure.



DCE--Data Communication Equipment

Note that pins 4 and 5 (RTS/CTS) are usually omitted in a full-duplex connection.

The following figure shows a simple connection of a computer to a DCE device such as a modem.



DTE--Data Terminal Equipment

DCE--Data Communication Equipment

10.3 Bidirectional Capabilities

The INTERACTIVE UNIX System `asy` driver allows a single serial port to be configured for simultaneous dial-in and dial-out use. To accomplish this, devices that interlock to prevent simultaneous access to the port are created for each `asy` port. `uugetty` is no longer necessary.

Each port has the following devices associated with it. For each hardware port `tty0`*N* (i.e., `tty01`):

`ttyd`*N*	Used for dial-in. `/etc/getty` should be placed on this line for incoming calls.
`acu`*N*	Used for dial-out. This is the device name that should be used when setting up `/usr/lib/uucp/Devices` for `cu` and `uucico`.
`tty0`*N*	Used for directly-connected devices. This is intended for hardwired terminals, printers, and mice. This device does not use the carrier-detect handshaking line. This device should *not* be used for modems.

For more information about the `asy` driver and its minor numbers, refer to "The Asynchronous Port Driver" in section 8.4.

10.4 Setting Up Modems

Modems can be used to connect your computer to remote terminals and computers using ordinary telephone lines. Terminals attached to a modem can dial in to your system from a remote location, and the INTERACTIVE UNIX Operating System and other UNIX System machines with a modem can use `uucp` to exchange information with your computer. Configuring your system to support a modem is similar to configuring a terminal, but more complex. Each requires an entry to be made in the `/etc/conf/init.d/asy` file, but the entry can differ, depending on the intended use of the modem.

A modem can be configured to support either single-directional or bidirectional communications. Single-directional communication means that only incoming or outgoing calls are allowed through the modem, while bidirectional communications allow your system to receive incoming and initiate outgoing calls using the modem. Therefore, you must determine the desired type of communications.

☛ Your modem must have auto-dial capability if you want outgoing or bidirectional communications.

If only incoming communication is desired, the calling terminal or computer must be issued a `getty` process. Therefore, an entry similar to the following would be added to `/etc/inittab`:

```
    I00:23:respawn:/etc/getty ttyd0 1200
```

Single-directional, outgoing communication requires that neither `getty` nor `uugetty` be attached to the port. In this case, the *action* field (field number 3) should be set to `off` instead of `respawn`.

In addition to the `/etc/inittab` entry, the `/usr/lib/uucp/Devices` file must be updated to configure the system to support modem dialing. Refer to section 11, "BASIC NETWORKING ADMINISTRATION," for a description of the `/usr/lib/uucp/Devices` file.

10.4.1 Initial Modem Installation

Initial modem setup occurs the first time the modem is installed. You should:

1. Make sure the modem power switch is off.

2. Set up the modem option switches, if any.
3. Physically connect the modem to the computer and the telephone line (i.e., attach the cables).
4. Plug the modem in.
5. Turn on the power to the modem.
6. Configure the kernel as described in section 7, "USING `kconfig` TO TAILOR YOUR SYSTEM KERNEL."
7. Return to this section after you have set up the serial port.

The software commands necessary to configure the modem are then executed. These commands are sent to the modem, so the modem must be connected to the computer and the power to the modem must be on. In addition, a number of configuration files on the computer must be modified. These steps may take a few moments.

The following sections contain switch and register settings for some popular modems. Consult the appropriate section for your particular type of modem. If there is no entry for your specific modem, use another type as a guide, and refer to the modem manufacturer's documentation for your specific settings.

If you follow the instructions in sections 10.4.2 and 10.4.3 to configure a COM port, your port will have bidirectional capabilities.

If your modem is configured using software registers instead of DIP switches, follow the directions in section 10.4.3 on dial-out use to set up a "direct" outgoing connection. Then use `cu` to connect to the modem (i.e., `cu -ltty0n`). You can then talk directly to the modem and send the appropriate `AT` codes to set the modem registers.

If you have a DOS communications program that you are familiar with, such as "kermit" or the BASICA "comm" program, you may prefer to set the modem software registers using that.

10.4.2 Configuring Dial-In Modem Lines

To set up dial-in lines, you need to inform the `init` process which devices should run `gettys`. Edit `/etc/conf/init.d/asy` to add the correct `getty` lines for each modem. For example:

```
# Hayes-compatible 1200 baud modem on ttyd0
00:2345:respawn:/etc/getty -t 60 ttyd0 d1200
# Hayes-compatible 2400 baud modem on ttyd1
01:2345:respawn:/etc/getty -t 60 ttyd1 D2400
# Telebit Trailblazer on ttyd2
02:2345:respawn:/etc/getty -t 60 ttyd2 TB19200
```

The `getty` line contains information on the status of the port, the timeout value for the modems, the port to spawn the `getty` on, and the type of `getty` to be used, as named in `/etc/gettydefs`.

If the field marked `respawn` is changed to `off`, the port will be disabled and will not allow logins. Refer to section 10.4.9, "Commented `gettydefs` Listing," to find the proper `gettydef` for the modem being installed. See *getty*(1M) for a further explanation of the options to `/etc/getty`.

After `/etc/conf/init.d/asy` has been configured, a new `/etc/inittab` file must be built. Do the following:

1. Log in as or `su` to `root` and type:

   ```
   # /etc/conf/bin/idmkinit
   ```

   to build a new `inittab` file from the files in `/etc/conf/init.d`.

2. Type:

   ```
   # mv /etc/inittab /etc/inittab.save
   ```

   to save your old `/etc/inittab` file.

3. Type:

   ```
   # mv /etc/conf/cf.d/inittab /etc/inittab
   ```

   to install the new `inittab` file.

4. When this is done, the changes can be tested by running:

   ```
   # telinit q
   ```

Editing `/etc/conf/init.d/asy` preserves the changes through future kernel builds. Changes made by directly editing `/etc/inittab` will be lost, as this file is overwritten the first time a new kernel is booted.

If you are installing an external modem with a DTR indicator light, it should turn on. This indicates that `/etc/init` has spawned a `getty` on the modem and is waiting for a caller. If the RD (Receive Data) and/or SD (Send Data) lights on the modem panel start flashing quickly at this point, it indicates that the modem has not been configured properly or the cable is not wired correctly. If this occurs, restore the old inittab file and cause init to reread the inittab file by typing:

```
# mv /etc/inittab.save /etc/inittab
# telinit q
```

10.4.3 Configuring Dial-Out Modem Lines

To configure a dial-out line the Dialers and Devices files in /usr/lib/uucp must be modified.

1. Edit /usr/lib/uucp/Dialers to add the Dialers entry for your particular modem as indicated in the appropriate modem section in section 11. There may already be an entry for your modem in the file. The entries in section 11 have been tested with the INTERACTIVE asy drivers and are known to work.

2. For each device you are configuring for a dial-out line, edit /usr/lib/uucp/Devices to add one direct entry for configuring (e.g., changing the modem software registers) and testing and one or more acu-style entries.

   For example:

```
# Hayes SmartModem 1200 on acu0 (direct access via tty00)
Direct  tty00   -       Any     direct  \D
ACU     acu0    -       300     hayes1200 \T
ACU     acu0    -       1200    hayes1200 \T

# Hayes SmartModem 2400 on acu1 (direct access via tty01)
Direct  tty01   -       Any     direct  \D
ACU     acu1    -       300     hayes2400       \T
ACU     acu1    -       1200    hayes2400       \T
ACU     acu1    -       2400    hayes2400       \T

# Telebit Trailblazer on acu2 (direct access via tty02)
Direct  tty02   -       Any     direct  \D
ACU     acu2    -       300     tb300 \T
ACU     acu2    -       1200    tb1200 \T
ACU     acu2    -       2400    tb2400 \T
ACU     acu2    -       9600    tbfast \T
ACU     acu2    -       19200   tbfast \T
```

10.4.4 Troubleshooting Modem Installation

This section discusses problems that may occur during installation and suggests solutions.

Modem Transmit and Receive lines are flashing – no connection established.

Modem is in “verbose” mode. Check the “verbose” switch setting or register value, as documented for your particular modem.

Modem transmits data as soon as `getty` *is enabled.*

Modem has DCD forced high, so the system thinks a caller has dialed in. If the command `ps -eaf` shows the `TTY` as anything except ? before someone calls in, then DCD is somehow forced high. This is typically configured in the modem, or the cable may have the DCD pin tied high.

`/etc/init` *prints the message* `getty respawning too rapidly`.

Several problems can trigger this message. You may not have specified the correct device name in `inittab` or `/etc/conf/init.d/asy`, the device may not be configured into kernel properly (check `/etc/conf/sdevice.d/asy`), or the device may not be found by the kernel (check jumper settings for I/O address or possible addressing conflict, e.g., overlapping addresses with some other board).

The modem answers the phone but gives no `login` *prompt.*

`getty` is either somehow misconfigured or the kernel is configured for wrong interrupt line. Check to see that `/etc/conf/sdevice.d/asy` specifies the correct IRQ (or `/etc/conf/pack.d/asy/space.c` has been properly edited for shared interrupts). Another possible cause is that you are attempting to share interrupts but the hardware cannot handle it. Try again with only one device on the interrupt line.

The modem answers the phone, but garbage characters come out.

Try pressing BREAK (or `~%b` if using `cu`) or ENTER several times to get `/etc/getty` to cycle through baud rates. This is normal behavior if you connected at other than the primary baud rate. If that does not work, check that the `gettydefs` entry specified in the `inittab` file cycles through the baud rate at which you connected.

If some characters appear to be correct, check that your character size (data bits) and parity agree. If the standard `gettydefs` entries are being used, you need 7 data bits, even parity. If you are using the 8-bit `gettydefs` entries, be sure to use 8 bits and no parity.

The modem answers the phone, the user can log in and work, but the modem does not hang up when the user logs off.

There are several possibilities.

- The modem may be ignoring DTR. Check the modem switches or registers. The modem should be set to drop the line and reset itself when DTR is dropped.
- The cable may be forcing DTR to always be high. In this case, the modem may also auto-answer even when the system is not in multi-user mode. Check that DTR goes low when `/etc/getty` is not running (i.e., change `respawn` to `off` in `/etc/inittab`).
- The "hang up on close" field is not set. Add the `HUPCL` flag to the associated `/etc/gettydefs` entry.

UUCP gives `no devices available` *message.*

- `getty` could be getting the device open without carrier. Check the cable and registers.
- A direct line could be open.
- Is there a mouse attached using this line?
- The entry in `/usr/lib/uucp/Devices` could be incorrect.

UUCP or `cu` *will not dial out properly.*

- The modem may be configured incorrectly. Check the switch settings and modem register settings.
- `/usr/lib/uucp/Devices` may specify an incorrect `Dialers` entry. See the examples in section 11.

  As a debugging tool, try using the following command, which attempts to set up a `uucp` connection to a designated site. The command is set at a debugging level (`-xN`) that shows the actions of the process:

```
# /usr/lib/uucp/uucico -r1 -x6 -s<hostname>
```

To check the modem connection and modem operation, use the following command:

```
# cu -sspeed -ltty0N
```

This gives you direct access to the modem and the ability to give it "AT" codes directly.

10.4.5 Recommended Modem Settings

Some modems, such as the Hayes SMARTMODEM 1200*, are hardware-configured. They have switch settings that must be manually set in order for the modem to work correctly. Other modems are configured via software or nonvolatile memory.

Hardware-configured modems that have a carrier detect (CD) switch usually must have that switch set low or off.

10.4.6 Using a Hayes SMARTMODEM 1200

The switch setting values may be different on Hayes*-compatible modems. Check the documentation that accompanied your modem.

Recommended Switch Settings		
1	UP	Modem obeys DTR
2	UP	English response codes
3	UP	(Dialin) no result codes unless enabled via ATQ0
	DN	(Dialout only) send result codes
4	UP	Echo commands (optional)
5	UP	(Dialin) enable auto-answer
	DN	(Dialout) disable auto-answer
6	UP	Carrier detect shows true carrier state
7	DN	Single-line telephone
8	UP	"Smart mode" (command recognition) (HAYES)
	DN	Enable 1200 baud operation (QUBIE) (This setting varies from modem to modem. You may have to find the proper setting for your modem.)

Add the following entry to the `/usr/lib/uucp/Dialers` file:

```
##########################
#
#  Hayes Smartmodem 1200
#
##########################
hayes1200 =,-,  "" \M\pAT\r\dATQ0V1\r\c OK\r ATDT\T\r\m\c CONNECT
```

Add the following entries to the `/usr/lib/uucp/Devices` file:

```
# Hayes SmartModem 1200 on acuN (direct access via tty0N)
Direct  tty0N   -       Any     direct  \D
ACU     acuN    -       300     hayes1200 \T
ACU     acuN    -       1200    hayes1200 \T
```

10.4.7 Using a Hayes SMARTMODEM 2400 (Nonvolatile Settings)

These settings should be set using the DOS "comm" program or the `cu -ltty0`*N* command. Issue the commands listed in the first column of the following table:

Recommended Settings	
AT&F	Reset to factory settings
ATQ1	No result codes (unless explicitly enabled using ATQ0)
ATM0	[Optional – speaker always off]
ATE1	Echo command characters
ATV1	English result codes
ATX4	Extended status (Allows Hayes to detect dial tone and busy signal)
AT&C1	Indicate true carrier state
AT&D3	Hangup and reset to saved values when DTR is dropped
ATS0=1	[set this for dial-in use only] Answer phone after 1 ring. You may want to increase this value if the phone line will be used for both voice and computer communications.
AT&W	Save settings to nonvolatile memory

Add the following entry to `/usr/lib/uucp/Dialers`:

```
##########################
#
#  Hayes Smartmodem 2400
#
##########################
hayes2400 =,-,  "" \M\pAT\r\dATQ0V1\r\c OK\r ATDT\T\r\m\c CONNECT
```

Add the following entries to `/usr/lib/uucp/Devices`:

```
# Hayes  SmartModem 2400 on acu1 (direct access via ttyON)
Direct   ttyON     -      Any    direct  \D
ACU      acuN      -      300    hayes2400        \T
ACU      acuN      -      1200   hayes2400        \T
ACU      acuN      -      2400   hayes2400        \T
```

10.4.8 Telebit TrailBlazer Suggested Configurations (Nonvolatile Settings)

The following register settings are known to work for the Telebit* TrailBlazer* Plus at ROM revision level "BA4.00," as reported by the ATN? and ATI3 commands.

ATE1	Enable modem to echo characters while in command mode (default).
ATF1	Disable echoing of data stream (default).
ATM1	Enable speaker during dial and connecting steps only (default). (You may wish to completely disable the speaker via "M0" instead.)
ATQ4	Modem will not report result codes unless connection was initiated locally via an ATD or ATA command. This is an important register setting.
ATV1	Send result codes as English words instead of numeric codes (default).
ATX3	Enable PEP and MNP extended result codes.

"S" register settings (only non-default settings are listed here):

ATS00=001	Answer on the first ring. Set this to 0 if dial-in mode is not desired. You may want to increase this value if the phone line will be used for both voice and computer communications.
ATS51=254	Allow the interface speed to automatically match that of the modem connection. Also configures incoming PEP-mode calls to use 19200 baud. (Note that early versions of Telebit's documentation overlooked this value for s51. This setting behaves exactly as the documented value of 255 except that 254 uses an interface of 19200 baud instead of 9600 baud.)

ATS52=002	Disconnect modem and reset to stored parameters when DTR is dropped by the host system.
ATS53=002	DCD and DSR are on only when a carrier is detected.
ATS54=003	Break signals are passed in-line in the data stream.
ATS58=000	No flow control is used between modem and host when in PEP mode. (Because the interface speed should match the modem speed, this setting turning off flow control should not matter.)
ATS92=001	When answering the phone, issue PEP tones *after* the normal 300/1200/2400 sequence so that non-PEP callers do not get confused.
ATS110=001	Use data compression in PEP mode if remote also has it enabled.
ATS111=030	Use UUCP "g" protocol spoofing in PEP mode.
AT&W	Save the EEPROM configuration. (Remember to issue this command to save the settings.)

Other registers that may be of interest are:

S61	Modem volume control
S95	MNP operating mode

Add the following entries to `/usr/lib/uucp/Dialers`:

```
##########################
#
# Telebit TrailBlazer
#
##########################
tb300  =W-, "" A\pA\pA\pA\pA\pT OK ATx3s0=0s64=1s50=1DT\T NNECT
tb1200 =W-, "" A\pA\pA\pA\pA\pT OK ATx3s0=0s64=1s50=2DT\T NNECT\s1200
tb2400 =W-, "" A\pA\pA\pA\pA\pT OK ATx3s0=0s64=1s50=3DT\T NNECT\s2400
tbfast =W-, "" A\pA\pA\pA\pA\pT OK ATx3s0=0s64=1s50=255DT\T FAST-\c-FAST
```

Certain revisions of the TrailBlazer will do better with the following:

```
tb300  =W-, "" A\pA\pA\pA\pA\pT OK ATx3s0=0s64=1s50=0DT\T NNECT
tb1200 =W-, "" A\pA\pA\pA\pA\pT OK ATx3s0=0s64=1s50=0DT\T NNECT\s1200
tb2400 =W-, "" A\pA\pA\pA\pA\pT OK ATx3s0=0s64=1s50=0DT\T NNECT\s2400
tbfast =W-, "" A\pA\pA\pA\pA\pT OK ATx3s0=0s64=1s50=255DT\T FAST-\c-FAST
```

Add the following entries to `/usr/lib/uucp/Devices`:

```
# Telebit TrailBlazer on acuN (direct access via tty0N)
Direct tty0N  -    Any   direct \D
ACU    acuN   -    300   tb300 \T
ACU    acuN   -    1200  tb1200 \T
ACU    acuN   -    2400  tb2400 \T
ACU    acuN   -    9600  tbfast \T
ACU    acuN   -    19200 tbfast \T
```

10.4.9 Commented gettydefs Listing

This section shows some sample gettydefs entries. Rather than directly editing the /etc/gettydefs file, you can use the sysadm program to initialize a modem.

```
# This sequence is for a Telebit TrailBlazer modem.  It cycles
# through 19200, 2400, and 1200 baud.  It sets the line for the
# "traditional" UNIX System values of 7 data bits, even parity.
#
# To enable a getty on ttyd0 using this sequence, specify:
#
# answering initially at 19200 -- "/etc/getty -t 60 /dev/ttyd0 TB19200"
# answering initially at 2400  -- "/etc/getty -t 60 /dev/ttyd0 TB2400"
# answering initially at 1200  -- "/etc/getty -t 60 /dev/ttyd0 TB1200"
#

TB19200# B19200 HUPCL OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY
PARENB ISTRIP ECHO ECHOE ECHOK ICANON ISIG CS7 CREAD # B19200
HUPCL OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY PARENB ISTRIP
ECHO ECHOE ECHOK ICANON ISIG CS7 CREAD #login: #TB2400

TB2400# B2400 HUPCL OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY
PARENB ISTRIP ECHO ECHOE ECHOK ICANON ISIG CS7 CREAD # B2400
HUPCL OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY PARENB ISTRIP
ECHO ECHOE ECHOK ICANON ISIG CS7 CREAD #login: #TB1200

TB1200# B1200 HUPCL OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY
PARENB ISTRIP ECHO ECHOE ECHOK ICANON ISIG CS7 CREAD # B1200
HUPCL OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY PARENB ISTRIP
ECHO ECHOE ECHOK ICANON ISIG CS7 CREAD #login: #TB19200


#
# This sequence is also for the Telebit, the only difference
# being that each entry specifies 8 data bits, no parity, for
# use by most PC users. (INTERNATIONAL USERS: remove the
# "ISTRIP" keyword to allow 8-bit input.)
#

TB8-19200# B19200 HUPCL OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY
ISTRIP ECHO ECHOE ECHOK ICANON ISIG CS8 CREAD # B19200 HUPCL
OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY ISTRIP ECHO ECHOE
ECHOK ICANON ISIG CS8 CREAD #login: #TB8-2400

TB8-2400# B2400 HUPCL OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY
ISTRIP ECHO ECHOE ECHOK ICANON ISIG CS8 CREAD # B2400 HUPCL
OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY ISTRIP ECHO ECHOE
ECHOK ICANON ISIG CS8 CREAD #login: #TB8-1200

TB8-1200# B1200 HUPCL OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY
ISTRIP ECHO ECHOE ECHOK ICANON ISIG CS8 CREAD # B1200 HUPCL
OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY ISTRIP ECHO ECHOE
ECHOK ICANON ISIG CS8 CREAD #login: #TB8-19200


#
# This sequence is for 2400 baud modems.  It cycles through
# 2400, 1200, and 300 baud.  It sets the line to 7 data bits
# and even parity.
#
```

```
# To enable a getty on ttyd0 using this sequence, specify
# answering initially at 2400 -- "/etc/getty -t 60 /dev/ttyd0 D2400"
# answering initially at 1200 -- "/etc/getty -t 60 /dev/ttyd0 D1200"
# answering initially at  300 -- "/etc/getty -t 60 /dev/ttyd0 D300"
#

D2400# B2400 HUPCL OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY
PARENB ISTRIP ECHO ECHOE ECHOK ICANON ISIG CS7 CREAD # B2400
HUPCL OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY PARENB ISTRIP
ECHO ECHOE ECHOK ICANON ISIG CS7 CREAD #login: #D1200

D1200# B1200 HUPCL OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY
PARENB ISTRIP ECHO ECHOE ECHOK ICANON ISIG CS7 CREAD # B1200
HUPCL OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY PARENB ISTRIP
ECHO ECHOE ECHOK ICANON ISIG CS7 CREAD #login: #D300

D300# B300 HUPCL OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY
PARENB ISTRIP ECHO ECHOE ECHOK ICANON ISIG CS7 CREAD # B300
HUPCL OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY PARENB ISTRIP
ECHO ECHOE ECHOK ICANON ISIG CS7 CREAD #login: #D2400


#
#  This is the eight data bits, no parity version of the 2400
#  baud sequence.
#  (INTERNATIONAL USERS: remove the "ISTRIP" keyword to allow
#  8-bit input.)
#

D8-2400# B2400 HUPCL OPOST ONLCR TAB3 BRKINT IGNPAR IXON
IXANY ISTRIP ECHO ECHOE ECHOK ICANON ISIG CS8 CREAD # B2400
HUPCL OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY ISTRIP ECHO
ECHOE ECHOK ICANON ISIG CS8 CREAD #login: #D8-1200

D8-1200# B1200 HUPCL OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY
ISTRIP ECHO ECHOE ECHOK ICANON ISIG CS8 CREAD # B1200 HUPCL
OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY ISTRIP ECHO ECHOE
ECHOK ICANON ISIG CS8 CREAD #login: #D8-300

D8-300# B300 HUPCL OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY
ISTRIP ECHO ECHOE ECHOK ICANON ISIG CS8 CREAD # B300 HUPCL
OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY ISTRIP ECHO ECHOE
ECHOK ICANON ISIG CS8 CREAD #login: #D8-2400


#
# This sequence is for 1200 baud modems.  It switches between
# 1200 and 300 baud only, using 7 data bits, even parity.
#
# To enable a getty on ttyd0 using this sequence, specify
# answering initially at 1200  -- "/etc/getty -t 60 /dev/ttyd0 d1200"
# answering initially at  300  -- "/etc/getty -t 60 /dev/ttyd0 d300"
#

d1200# B1200 HUPCL OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY
PARENB ISTRIP ECHO ECHOE ECHOK ICANON ISIG CS7 CREAD # B1200
HUPCL OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY PARENB ISTRIP
ECHO ECHOE ECHOK ICANON ISIG CS7 CREAD #login: #d300

d300# B300 HUPCL OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY
PARENB ISTRIP ECHO ECHOE ECHOK ICANON ISIG CS7 CREAD # B300
HUPCL OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY PARENB ISTRIP

ECHO ECHOE ECHOK ICANON ISIG CS7 CREAD #login: #d1200
```

```
#
# This is the eight data bits, no parity version of the
# 1200/300 baud sequence.
# (INTERNATIONAL USERS: remove the "ISTRIP" keyword to allow
# 8-bit input.)
#
d8-1200# B1200 HUPCL OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY
ISTRIP ECHO ECHOE ECHOK ICANON ISIG CS8 CREAD # B1200 HUPCL
OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY ISTRIP ECHO ECHOE
ECHOK ICANON ISIG CS8 CREAD #login: #d8-300

d8-300# B300 HUPCL OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY
ISTRIP ECHO ECHOE ECHOK ICANON ISIG CS8 CREAD # B300 HUPCL
OPOST ONLCR TAB3 BRKINT IGNPAR IXON IXANY ISTRIP ECHO ECHOE
ECHOK ICANON ISIG CS8 CREAD #login: #d8-1200
```

10.5 The Default Hardware Configuration

The INTERACTIVE UNIX Operating System supports both AT-based systems and the Micro Channel architecture (PS/2) systems. The kernel delivered with the INTERACTIVE UNIX Operating System is capable of supporting the following default hardware configurations:

AT 386 systems assume:*

- An AT keyboard
- An IBM AT or AT-compatible fixed disk controller card or an AT SCSI host and floppy disk controller card
- A monochrome, color, EGA, or VGA display controller
- Two serial communications ports (COM1, COM2)
- One parallel printer port
- 4 MB AT system RAM

Micro Channel (MCA) systems assume:
(supports the PS/2 Model 70 and PS/2 Model 80)

- An MCA keyboard
- An MCA floppy controller
- An MCA fixed disk controller
- A VGA display controller
- One parallel printer port
- Two serial communications ports

The default configuration of your machine may be different. Consult the documentation provided by your vendor to determine your default hardware configuration.

10.6 Peripheral Devices

One of the tasks a system administrator is responsible for is adding peripheral devices such as terminals, line printers, or a second fixed disk to the system. These peripheral devices are used to increase the number of users or disk space or to expand the overall capabilities of the system.

Many of the computers supported by the INTERACTIVE UNIX System are configured with one or two serial (RS-232) ports and one parallel (CENTRONICS) interface port. While the serial ports can be used to connect any RS-232 device to your computer, the parallel port can only be used with printers. Typical RS-232 devices include terminals, modems, line printers, and other computers.

A port can be either serial or parallel. A serial port, such as `COM1` or `COM2`, allows the transmission of data 1 bit at a time; a parallel port, such as `LPT1` or `LPT2`, allows the transmission of data 8 bits at a time.

Terminals must be connected to the system through serial ports. When the computer is powered up or rebooted, it is automatically initialized to support both the console and the remote terminals if the serial ports are appropriately set up. Printers, in contrast, can be either serial or parallel, depending on how they have been configured by the manufacturer.

The INTERACTIVE UNIX Operating System has a hardware configuration that typically has at least one parallel and two serial ports. With such a configuration you might, for example, connect two terminals and a parallel printer or one terminal, one serial mouse, and a parallel printer.

This section explains how to add RS-232 devices and parallel line printers to your system.

10.6.1 Setting Up an RS-232 Connection

When connecting an RS-232 device to your system, a connection must be made from your computer to the device either directly (hardwired) or through a modem. Once the connection is made, the system must be told what type of connection exists and what type of device is being connected. You will configure the RS-232 port when adding terminals, modems, and serial line printers.

10.6.2 RS-232 Direct Connection

The direct, hardwired connection between your computer and an RS-232 device depends on the type of device. Consult the hardware manufacturer's documentation to determine the specific type of cable and pin assignments needed. The following hints generally apply when connecting an RS-232 device to your computer:

- RS-232 cables should be less than 50 feet long. Cables longer than 50 feet may introduce line noise. In some cases, you can use a longer cable by reducing the line speed (baud rate) or by using low-capacitance cables in conjunction with signal boosters.
- An in-line adapter called a null modem adapter is usually required when connecting a terminal or computer directly to the serial port on your computer. Alternately, the cable itself may have been modified to function as a null modem.
- A null modem cable or adapter should not be used when connecting a modem to your computer.

10.6.3 Configuring Your Computer With Additional Terminals

A terminal is connected to the system through a *port* (where the terminal is actually plugged in) on the motherboard (main board) or on a multi-port adapter board on your system. This allows more than one person to use the computer at one time.

Asynchronous serial ports are also called *tty lines*. A tty line is a line that connects a terminal, modem, or other device to the main system and is associated with a specific port. A tty line is also referred to as a *com* (communication) line by some manufacturers.

Tty lines are either alphabetically ordered or sequentially numbered. For example, some of the tty lines included on your system are `tty00` and `tty01`.

Each time you add a new terminal to your system, you will need to configure it for your system, using the `sysadm` `Tty Management` menu under `Machine` on the bar menu. However, you will not need to install a device driver or reconfigure and rebuild your kernel if you are using a terminal configuration that is already supported. Most terminals that use one of the two standard serial ports of your computer are supported by the default configuration supplied by INTERACTIVE. Check the documentation supplied by your vendor to determine whether your terminal is supported by this configuration. If it is, you can proceed immediately to step 1 of the procedure outlined in this section to add your new terminal.

If you plan to add a terminal that is not supported by the serial ports on the main board of your CPU, you must first install the terminal adapter (possibly with a special device driver), then configure, build, and install a new kernel (see sections 7.3, 7.4, and 7.5). You should then follow the steps outlined below.

If you plan to use a multi-port serial expansion card to connect your terminals, you may need to install additional device drivers. Refer to the manufacturer's installation instructions supplied with your hardware.

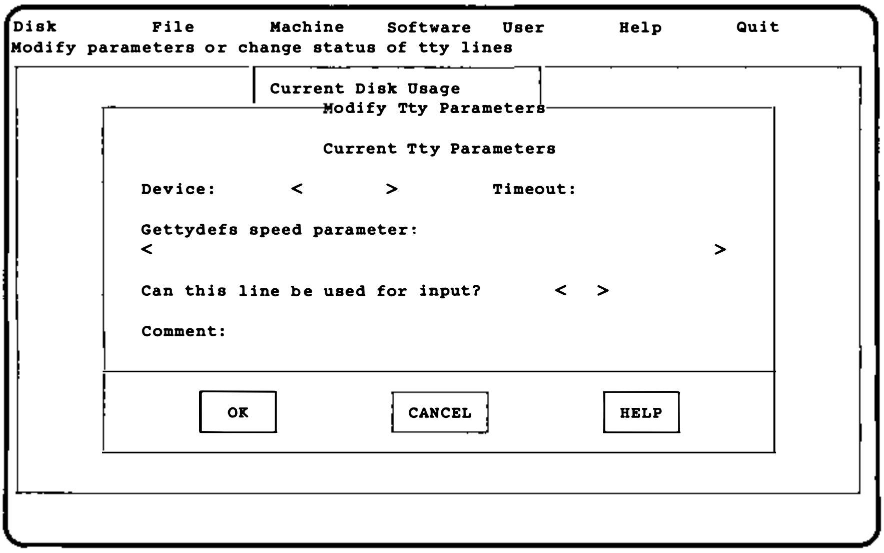
If you are unsure about whether your terminal type is supported on this system, check the `/usr/lib/terminfo/?` directory, where ? stands for the first character of the name of your terminal type. For example, the directory `/usr/lib/terminfo/a` will contain a file for each supported terminal whose name begins with the letter `a`. Alternatively, if you have installed the Terminal Utilities subset, you may use *infocmp*(1M) to check the system support for your terminal.

To install a new terminal, follow these procedures:

1. Refer to the manufacturer's information you received with your new terminal to determine the appropriate *baud rate* and how to set it. The baud rate is the speed at which data is to be transmitted to and from your terminal. Follow the manufacturer's instructions to set the baud rate on your terminal.

2. Refer to your hardware manufacturer's information to determine how the ports on your board are labeled and where to plug in the terminal. Plug the new terminal into the appropriate port on your main board or other serial board.

3. Log in as `sysadm` and select `Modify Tty Parameters` from the `Tty Management` menu under `Machine`, or type `sysadm chgtty` to access the `Modify Tty Parameters` form directly. The system displays a screen similar to this:



```
Disk          File        Machine     Software   User        Help        Quit
Modify parameters or change status of tty lines

                  Current Disk Usage
                         Modify Tty Parameters

                         Current Tty Parameters

     Device:          <        >         Timeout:

     Gettydefs speed parameter:
     <                                                        >

     Can this line be used for input?      <   >

     Comment:

          OK                 CANCEL                 HELP
```

Complete all fields on the form.

10.6.4 Requirements for Multi-User Operation

The following are requirements for the computer to support a remote terminal:

- The `TERM` shell variable must match the type of terminal attached to your computer. This ensures correct operation of screen-oriented applications, such as the `vi` editor. The default system profile (`/etc/profile`) automatically sets `TERM` to `at386`, corresponding to the computer console. At each remote terminal, you can set `TERM` from the command line by typing:

  ```
  TERM=terminal_name

  export TERM
  ```

  (If each terminal is used frequently, set up the `TERM` command line in your `.profile`. If you frequently use several different types of remote terminals, you can modify your `.profile` to reflect this also.)

  Alternatively, you can add a line to the `/etc/ttytype` file:

  ```
  terminal_name  device_name
  ```

  This will set the terminal to one particular type for anyone logging in on that tty line (device name). This is particularly useful for terminals that are directly connected; it probably should not be used for modem connections since you cannot necessarily predict the type of terminal used with a modem.

- The terminal must have the correct baud rate set.
- The remote terminal will probably require a null modem cable or adapter if directly linked to your computer (rather than through a modem over a telephone line).

10.6.5 Installing Software Support for Additional Terminals

Some of the software included in the Base operating system defines the terminals you can add to your computer. This group of terminal characteristics is known as the Terminal Information Library (`terminfo`). Each entry in the library represents one supported terminal.

By default, the only supported terminals are: the console (`AT386`), the DEC* VT100* (`vt100`), and all ANSI-compatible terminals (`ansi`).

If you are adding a different terminal to your computer, you must add the additional terminal information. These entries are packaged separately in the Terminal Utilities optional subset. The Terminal Utilities subset is used to install the `terminfo` database. It has a separate installation procedure that involves selecting a terminal type from a list.

10.7 Configuring and Maintaining a Serial Line Printer

A printer is added to the system using the `sysadm lpmgmt` command. Printers can be either serial or parallel. A serial printer must be plugged into a serial port, such as `COM1` or `COM2`; a parallel printer must be plugged into a parallel port, such as `LPT1` or `LPT2`. Printers can be associated with tty lines, just as terminals are.

The system supports a set of default printer types. However, the printers shown in this section are only examples. Your system may be configured to support additional or different printers. Before you attempt to install a new printer, refer to the documentation provided by your vendor or to your hardware manufacturer's instructions. Refer to section 7 in "System Administration for New Users of the INTERACTIVE UNIX Operating System" for information about using `sysadm lpmgmt` to add a printer and refer to section 9 in this document for information about the LP Print Service.

10.8 Configuring Other Directly Connected Devices

Other devices, such as another computer, can be connected to your computer using an RS-232 cable. Configuring your system to support such a connection is similar to configuring your system to support a modem. The same type of entry must be made in the `/etc/inittab` file to allow bidirectional or single-directional communication.

The difference lies in the `/usr/lib/uucp/Devices` file entry. Section 11, "BASIC NETWORKING ADMINISTRATION," describes the `/usr/lib/uucp/Devices` file and the proper entries for directly connected devices.

11. BASIC NETWORKING ADMINISTRATION

11.1 Introduction

This section describes the administration of the Basic Networking Utilities (BNU). It allows you to communicate with other INTERACTIVE UNIX System computers using either dial-up or hardwired communication lines. The BNU comes as part of the base INTERACTIVE UNIX Operating System in the Basic Networking optional subset.

The installation script for the Base system is normally used to select the type(s) of modems that you will be using. However, if you have not made a selection, refer to section 10.6.1, "Setting Up an RS-232 Connection." The basic instructions for setting up your computer to communicate with other computers is provided there. This includes connecting your computer to a modem so you can send electronic mail. Once a particular modem has been selected, refer to section 10, "ADDING MODEMS, PRINTERS, AND OTHER SERIAL DEVICES," for additional details.

Additional information on using Basic Networking can be found in Chapter 9, "Communication Tutorial," of the *User's Guide.*

Basic Networking is complex; the documentation included in this section will cover only the information most important for you.

11.2 Terms You Need to Know

The following list contains some terms used in Basic Networking and a brief description of each item:

local machine	Refers to the machine on the "near" end of a communication link, normally your computer.
remote machine	Refers to a machine on the "far" end of a communication link, normally a machine to which your computer talks.
active machine	A machine that has Basic Networking and the hardware required to establish communication links (e.g., Auto Dial Modem).
passive machine	A machine that has Basic Networking but does not have the hardware required to establish communication links.

network	A group of machines set up to exchange information and resources.
node	A terminating point (machine) on a network.
UUCP	Indicates a group of programs and files that allow systems to send or copy information from one system to another. UUCP means "UNIX System-to-UNIX System copy." In general, it refers to Basic Networking with the exception of the `cu` and `ct` programs. If "uucp" (written in lowercase) is used in text with constant width type (`uucp`), it refers specifically to the `uucp` program or login ID.

11.3 Overview of Basic Networking

Basic Networking allows machines using the INTERACTIVE UNIX Operating System or any UNIX System to communicate with one another. In general, Basic Networking allows you to do the following:

- Transfer files and send electronic mail to other UNIX System machines as background processes
- Interactively communicate with other UNIX System machines and, in some cases, non-UNIX System machines
- Execute restrictive subset commands on a remote machine without logging in
- Call a remote terminal and allow the user of that terminal to log in on your system

11.4 Hardware Requirements

Before your computer can communicate with a remote machine, a communication link must be established to the remote machine. There are two types of hardware used to establish a communication link to another machine.

The first is a direct link from a serial port on the computer to a serial port on the other machine. This type of connection is useful when two machines communicate with each other on a regular basis. Even though the RS-232 standard recommends that direct links be limited to 50 feet or less, two machines may be separated by several hundred feet provided that noise on the direct link does not become a problem. If noise becomes a problem or greater distance is needed between the two machines, the transfer rate may need to be decreased or limited distance modems placed at each end of the connection.

The second type of communication link uses the telephone network. In this type of link, the machine that establishes the connection (local machine) must have an Automatic Call Unit (ACU). The ACU dials the specified telephone number upon request from Basic Networking. The called (remote) machine must have a telephone modem capable of answering incoming calls so that other machines can contact it through the telephone network. The INTERACTIVE UNIX Operating System supports a number of automatic dial modems as ACUs. Refer to section 10, "ADDING MODEMS, PRINTERS, AND OTHER SERIAL DEVICES" for details.

11.5 The Basic Networking Software

Basic Networking is composed of software programs, daemons (background routines), and a supporting database. The supporting database contains support files that store information such as telephone numbers, location of the devices (hardware) used to establish links, and security restrictions. The software programs and a skeleton database are supplied in Basic Networking.

11.5.1 The Directories and Their Purpose

There are several directories that contain the programs and support files of Basic Networking. Some of these directories are unique to Basic Networking, while others are also common to the INTERACTIVE UNIX Operating System. The directories used by Basic Networking are:

`/usr/bin`	This directory is used by the INTERACTIVE UNIX Operating System and by Basic Networking to store executable programs.
`/usr/lib/uucp`	This directory is the "home" directory for the `uucp` administrative login. It contains the files of the supporting database and some executable programs.
`/usr/spool/locks`	This directory contains the lock (LCK) files for the Basic Networking

	hardware devices. Lock files prevent duplicate conversations and multiple attempts to use the same device.
`/usr/spool/uucp`	This directory is the "spool directory" for work that is to be processed by Basic Networking. It contains a tree-like structure of subdirectories associated with remote machines that your computer communicates with or has communicated with recently. These subdirectories are also used for administrative purposes such as storing log and status information.
`/usr/spool/uucppublic`	This directory is the "public" directory for UUCP transfers. The public directory is used to store files that have been sent to your computer. Some remote machines may be restricted to placing files in this directory, while others may have permission to place files elsewhere.

11.5.2 The Software Programs and Their Purpose

There are several types of programs associated with Basic Networking. Some are used by regular users to transfer data and obtain status information, while others are used for administration purposes or are executed internally. The following paragraphs contain a brief description of the programs and their purpose.

11.5.2.1 User Programs.

`cu`	Connects your computer to a remote machine and allows you to be logged in on both machines at the same time. This allows you to transfer files or execute commands on either machine.
`ct`	Connects your computer to a remote terminal and allows log in from that terminal. The user of the remote terminal may call into the computer and request that the computer call the remote terminal back. In this case, the computer drops the initial

link so that the modem will be available when it is called back.

`uucp`	Allows you to transfer files between UNIX Systems and performs all of the preliminary work to allow you to send files to remote machines. It creates work files that contain the instructions for transferring the queued file(s). Depending on the options specified, it may make a copy of the file to be transferred in the spool directory. These files are called data files. Once the work and data files have been created, `uucp` calls the `uucico` daemon that attempts to contact the remote machine to deliver the files.
`uuto`	This program works very similarly to the `uucp` program. In fact, it calls the `uucp` program to create work and data files. The main difference between `uuto` and `uucp` is the way the transferred files are placed on the remote machine. With `uucp`, you can specify a path name on the remote machine where you want the files to be placed. With `uuto`, all transferred files are placed in the directory `/usr/spool/uucppublic/receive`. See *uuto*(1C) for additional information.
`uupick`	When files are transferred to a machine using `uuto`, `uupick` can be used to retrieve the files placed under `/usr/spool/uucppublic/receive`.
`uux`	This program creates work files, data files, and execute files for executing commands on a remote machine. The work file contains the same information as files created by `uucp` and `uuto`. The execute files contain the command string to be executed on the remote machine and a list of the data files. The data files are those files required for the command execution.
`uustat`	This program displays status information for requested transfers (`uucp`, `uuto`, or `uux`). It also provides a means of controlling queued transfers.

11.5.2.2 Administrative Programs.

`uulog`	This program displays the contents of a specified machine's log file. Individual log files are created for

	each remote machine that your computer communicates with using the `uucp`, `uuto`, and `uux` programs.
`uucleanup`	This program has several functions associated with the cleanup of the spool directory. It is usually executed out of the `uudemon.cleanu` shell script that is started by `cron`. See *cron*(1M) for additional information.
`Uutry`	This program is a shell script used to test call processing capabilities with a moderate amount of debugging. It invokes the `uucico` daemon to establish the communication link between your computer and the specified machine.
`uucheck`	This program checks for the presence of Basic Networking directories, programs, and support files. It is also capable of checking certain parts of the `Permissions` file.

11.5.2.3 Internal Programs.

`uugetty`	This program is not needed for most purposes. It is very similar to the `getty` program except it permits a line (port) to be used in both directions. The `uugetty` program allows users to log in on your computer; if the line is not in use, it will allow `uucico`, `cu`, or `ct` to use it for dialing out. If one of these programs attempts to dial out when the line is busy, `uugetty` denies the requester permission and echoes a message indicating that the device is unavailable. The `uugetty` is executed as a function of the `init` program.

11.5.3 The UUCP Daemons and Their Purpose

Basic Networking contains three daemons (routines that run as background processes to handle file transfers and command executions).

`uucico`	This daemon is referred to as the transport program for UUCP requests. It selects the device used for the link, establishes the link to the remote machine, performs the required login sequence, and performs permission checks. It also transfers data and execute

files, logs results, and notifies specified users of transfer completions via `mail`. When the local `uucico` daemon calls a remote machine, it "talks" to the `uucico` daemon on the remote machine during the session. The `uucico` daemon is executed by several methods. It is started by the `uucp`, `uuto`, and `uux` programs to contact the remote machine after all the required data, work, and/or execute files have been created. It is also started by the `uusched` and `Uutry` programs.

`uuxqt` This daemon is the execution program for remote execution requests. It searches the spool directory for execute files (`X.`) that have been sent from a remote machine. When an `X.` file is found, `uuxqt` opens it to get the list of data files required for the execution. It then checks to see if the required data files are available and accessible. If the files are present and can be accessed, `uuxqt` checks the `Permissions` file to verify that it has permission to execute the requested command. The `uuxqt` daemon is executed out of the `uudemon.hour` shell script that is started by `cron`.

`uusched` This daemon schedules the queued work in the spool directory. Before starting the `uucico` daemon, `uusched` randomizes the order in which remote machines will be called. The `uusched` is executed out of a shell script called `uudemon.hour` that is started by `cron`.

11.5.4 The Supporting Database Files and Their Purpose

As mentioned earlier, several of the Basic Networking programs require information contained in support files. These support files are located in the `/usr/lib/uucp` directory. The `cu`, `ct`, `uucico`, and `uuxqt` programs require supporting information from the following files:

`Devices` This file contains information concerning the location and line speed of the automatic call unit, direct links, and possibly network devices.

`Dialers`	This file contains character strings required to negotiate with network devices (automatic calling devices) in the establishment of connections to remote computers (non-801-type dialers).
`Systems`	This file contains information needed by the `uucico` daemon (and possibly the `cu` program) to establish a link to a remote machine. It contains information such as the name of the remote machine, the name of the connecting device associated with the remote machine, when the machine can be reached, the telephone number, the login ID, the password, and so on.
`Dialcodes`	This file contains dial-code abbreviations that may be used in the phone number field of `Systems` file entries.
`Permissions`	This file defines the level of access granted to machines when they attempt to transfer files or remotely execute commands on your computer.

There are several other files that may be considered part of the supporting database, but these files are not directly related to the process of establishing a link and transferring files. For this reason, discussion of these files is reserved for section 11.7, "Administration."

11.6 How Basic Networking Operates

There are five programs that allow your computer to communicate with remote machines. The following paragraphs briefly describe what happens when you execute these programs.

11.6.1 `ct` *Program – Connect a Terminal*

The `ct` program instructs your computer to initiate a call to a remote terminal and issue a `getty` to that remote terminal. The `ct` command line must contain the telephone number of the remote terminal. Of course, the remote terminal must be attached to a modem that will automatically answer the call.

When the `ct` command line is issued, the `ct` program searches for an automatic dialer in the `Devices` file with a transfer rate that matches what was specified in the command line. If no transfer rate was specified, it defaults to 1200 bps. When `ct` finds the dialer to be used, it attempts to dial the telephone number specified in the command line. If no dialer is available, `ct` asks if it should wait for an available dialer and, if so, how many minutes it should wait. An option is available to override this dialogue. When the modem at the remote terminal answers the call from your computer, it is issued a `getty` (login) process. At this point, the user at the remote terminal may attempt to log in.

The user at a remote terminal can call your computer, log in, and request that the computer call the remote terminal back using the `ct` command. If this scenario is used, the remote user issues a `ct` command, and the link from the remote terminal is dropped. After `ct` finds an available dialer in the `Devices` file, it calls the remote terminal back.

11.6.2 `cu` *Program – Call a UNIX System*

The `cu` command enables you to call another machine and log in as a remote user. The telephone number or node name of the remote machine is required in the command line. If the telephone number is specified, it is passed on to the automatic dial modem. If a system name is specified, the telephone number is obtained from the associated `Systems` file entry. If an automatic dial modem is not used to establish the connection, the line (port) associated with the direct link to the remote machine can be specified in the command line.

If an automatic dial modem is used, the `cu` program will search for an automatic dialer in the `Devices` file with a transfer rate that matches what was specified in the command line. If no speed is specified, the first dialer listed (if available) will be used regardless of its transfer rate. After the link has been established and you have successfully completed the login process, you will be logged in on both computers. This allows you to execute commands on either computer and/or transfer ASCII coded files from one computer to another. After you have terminated the connection, you will still be logged in on your computer (calling computer). This command can only be executed by an active computer.

11.6.3 `uucp` *Program – UNIX System-to-UNIX System Copy*

The `uucp` command allows you to transfer file(s) to a remote computer without knowing any details of the connection. All that you are required to know is the name of the remote computer and possibly the login ID of the remote user to whom the file(s) is being sent. The details of the connection are kept in the `Systems` file.

When you enter a `uucp` command, the `uucp` program creates a work file and possibly a data file for the requested transfer. The work file contains information required for transferring the file(s). The data file is simply a copy of the specified source file. After these files have been created in the spool directory, the `uucico` daemon will start.

The `uucico` daemon attempts to establish a connection to the remote machine that is to receive the file(s). It first gathers the information required for establishing a link to the remote machine from the `Systems` file so that it knows what type of device to use in establishing the link. `uucico` then searches the `Devices` file for devices that match the requirements listed in the `Systems` file. After `uucico` has found an available device, it attempts to establish the link and log in on the remote machine.

When `uucico` logs in on the remote machine, it starts the `uucico` daemon on the remote machine. The two `uucico` daemons then negotiate the line protocol to be used in the file transfer(s). The local `uucico` daemon then transfers the file(s) to the remote machine, and the remote `uucico` places the file in the specified path name(s) on the remote machine. After your computer completes the transfer(s), the remote machine may send files that are queued for your computer. The remote machine can be denied permission to transfer these files with an entry in the `Permissions` file. If this is done, the remote machine must establish a link to your computer to perform the transfers. If the remote machine or the device selected to make the connection to the remote machine is unavailable, the request remains queued in the spool directory. Each hour, `cron` starts `uudemon.hour`, which in turn starts the `uusched` daemon. When the `uusched` daemon starts, it searches the spool directory for the remaining work files, generates the random order in which these requests are to be processed, and then starts the transfer process (`uucico`) described in the previous paragraphs.

The transfer process described generally applies to an active machine. An active machine (one with calling hardware and Basic Networking software) can be set up to "poll" a passive machine. A passive machine can queue file transfers (because it has Basic Networking software), but it cannot call the remote machine because it does not have the required hardware. The `Poll` file (`/usr/lib/uucp/Poll`) contains a list of machines that are to be polled in this manner. For additional information, refer to the discussion of the `Poll` file in section 11.8.6, "`Poll` File," and `uudemon.poll` in section 11.10.4, "`uudemon.poll`."

11.6.4 `uuto` *Program – Public UNIX System-to-UNIX System Copy*

The `uuto` program uses the `uucp` program to build work files and data files in the spool directory for requested transfers. The difference is that the `uuto` command does not allow you to specify a path name as a destination for the file. The `uuto` command automatically puts the file in a directory under `/usr/spool/uucppublic/receive`. Once the transfer is complete, mail is sent to the appropriate user indicating that a file has arrived and has been placed in the public area. That user can then use the `uupick` command to retrieve that file. The `uupick` command searches the public area for files destined to the user and allows the user to interactively delete, print, or move the file to a named directory.

11.6.5 `uux` *Program – UNIX System-to-UNIX System Execution*

The `uux` command allows commands to be executed on a remote machine. It gathers files from various computers, executes the specified command on these files, and sends the standard output to a file on the specified computer. Remote mail is implemented using the `uux` program, but its execution is embedded in the standard `mail` command. For security, many machines limit the list of commands that can be executed via `uux` to the default (receipt of mail).

When the `uux` command is issued, the `uux` program creates an execute (`X.`) file that contains the names of the files required for execution, your login name, the destination of the standard output, and the command to be executed. The `uux` command also creates work (`C.`) files that are used to gather the files required for execution. These files are then sent to the remote machine, along with the execute file, by the `uucico` daemon and placed in the remote spool directory.

Periodically, the `uuxqt` daemon on the remote machine is started to search for `X.` files in the spool directory. When it finds an `X.` file, the `uuxqt` daemon checks to see if all the required data files are available and accessible. It then checks the `Permissions` file to verify that the command(s) listed can be performed. After execution, `uuxqt` sends the standard output to a file on the specified computer.

11.7 Administration

The files and tasks associated with the operation of Basic Networking are discussed here. The amount of effort required to administer Basic Networking depends on the amount of traffic that enters or leaves your computer. For an average computer, little, if any, intervention is required by the automatic cleanup functions. A computer with a large amount of traffic may require more attention as problems arise.

As should be evident, the UUCP facilities make up the bulk of Basic Networking. These can generally be defined as all of the programs and support files in Basic Networking with the exception of the `ct` and `cu` programs.

11.7.1 Administrative Files

11.7.1.1 `TM.` *– Temporary Data File.* This data file is created under the spool directory (`/usr/spool/uucp/`*XXXX*) when receiving a file from another machine. The directory *XXXX* has the same name as the remote machine that is sending the file. The temporary data file name has the following format:

`TM.`*pid.ddd*

pid is a process ID and *ddd* is a sequential 3-digit number starting at zero.

After the entire file is received, the `TM.` file is moved to the path name specified in the command line. If the file was sent via the `uuto` program, the file is automatically moved to the public area. If processing is abnormally terminated, the `TM.` file may remain in the *XXXX* directory. This file should be periodically removed.

11.7.1.2 `LCK` *– Lock File.* The lock file is created in the `/usr/spool/locks` directory for each device in use. A lock file prevents duplicate conversations and multiple attempts to use the same calling device. The file name has the following format:

`LCK..`*str*

where *str* is either a device or computer name. The file may be left in the spool directory if runs abort (usually on computer crashes). The lock file will be ignored (reused) after the parent process is no longer active.

11.7.1.3 Work (C.) File. The work file is created in a spool directory when work (transfers or remote command executions) has been queued for a remote computer. The name has the following format:

`C.`*sysnxxxx*

where *sys* is the name of the remote computer, *n* is the ASCII character representing the grade (priority) of the work, and *xxxx* is the 4-character job sequence number assigned by UUCP. A work file contains the following information:

- Full path name of the file to be sent or requested
- Full path name of the destination or `~user/filename`

  ☛ The `~` is shorthand for `/usr/spool/uucppublic` and must be included if the full path name is not used.

- User login name
- List of options
- Name of associated data file in the spool directory (If the `-c` or `-p` option was specified, a dummy name [`D.0`] will be used.)
- Mode bits of the source file
- Remote user's login name to be notified upon completion of the transfer

11.7.1.4 Data (D.) File. The data file is created when it is specified in the command line to copy the source file to the spool directory. The file name has the following format:

`D.`*sysnxxxx*

where *sys* is the name of the remote computer, *n* is the character representing the grade (priority) of the work, and *xxxx* is the 4-character job sequence number assigned by UUCP. The 4-character job sequence number may be followed by a subjob number that is used when there are several `D.` files created for a work (`C.`) file.

11.7.1.5 Execute (X.) File. The execute file is created in the spool directory prior to remote command executions. The file name has the following format:

`X.`*sysnxxxx*

where *sys* is the name of the remote computer, *n* is the character representing the grade (priority) of the work, and *xxxx* is the 4-character sequence number assigned by UUCP.

The execute file contains the following information:

- Requester's login and computer name
- Name of file(s) required for execution
- Input to be used as the standard input to the command string
- Computer and file name to receive standard output from the command execution
- Command string
- Option lines for return status requests

11.7.1.6 Machine Log File. A log file is created for each remote machine with which your computer communicates. Each machine may have four log files, one for `uucico`, `uuxqt`, `uux`, and/or uucp requests, depending on the type of communication that has taken place. The log files are kept in the directory /usr/spool/uucp/.Log. Each day, these log files are combined and stored in the directory /usr/spool/uucp/.Old when uudemon.cleanu is executed. The combined files are kept 3 days before they are removed. If space is a problem, the administrator may consider reducing the number of days the files are kept by modifying the uudemon.cleanu shell file.

11.8 Supporting Database

The database that supports Basic Networking is composed of several support files. These support files contain information required by the uucico and uuxqt daemons during file transfers or remote command executions. All of the support files are located in the /usr/lib/uucp directory.

11.8.1 Devices *File*

The Devices file (/usr/lib/uucp/Devices) contains the information for all of the devices that may be used to establish a link to a remote machine. It contains information for both automatic call units, direct links, and network connections. Although provisions are made for several types of devices, only modems and direct links are supported.

This file works very closely with the Dialers, Systems, and Dialcodes files. It may be beneficial to become familiar with these files before attempting to understand the Devices file.

Each entry in the Devices file has the following format:

Type Line Line2 Class Dialer-Token-Pairs (DTP)

Each field (separated by a space) is defined in the following paragraphs.

- *Type*
  This field may contain one of five keywords:

Direct	This keyword indicates a direct link to another computer (for cu connections only).
ACU	This keyword indicates that the link to a remote computer is made through an automatic call unit (Automatic Dial Modem). This

	modem may be connected either directly to the computer or indirectly through a Local Area Network (LAN) switch.
Network	This keyword indicates that the link is established through a LAN switch where *Network* is replaced with either `micom` or `develcon`. These two LAN switches are the only ones that contain caller scripts in the `Dialers` file. Other switches may be used if caller scripts are constructed and placed in the `Dialers` file.
`Modem Control`	This keyword causes the device to be opened with `O_NDELAY` set (so the open does not hang waiting for carrier). After the open, `O_NDELAY` is cleared.
System-Name	This keyword indicates a direct link to a particular machine where *System-Name* is replaced by the name of the particular computer. This naming scheme is used to convey the fact that the line associated with this `Devices` entry is for a particular machine.

The keyword used in the *Type* field is matched against the third field of `Systems` file entries as follows:

```
Devices: ACU tty01,M - 1200 penril

Systems: eagle Any ACU 1200 3-2-5-1 ogin: nuucp ssword: Oakgrass
```

- *Line*
  This field contains the device name of the line (port) associated with the `Devices` entry. For instance, if the Automatic Dial Modem for a particular entry was attached to the `/dev/tty01` line, the device name would be `tty01`. The `,M` indicates that modem control is being used.

- *Line2*
  If the ACU keyword was used in the *Type* field and the ACU is an 801-type dialer, this field would contain the device name of the 801 dialer. It should be noted that 801-type ACUs do not contain a modem. Therefore, a separate modem is required and would be connected to a different line (defined in the *Line* field). This means that one line would be allocated to the modem and another to the dialer. Since the computer will not normally use this type of configuration, this field is ignored but must contain a pseudo entry as a placeholder (use a "-" as a placeholder).

- *Class*
  If an ACU keyword is used, this may be just the speed of the device. It may contain a letter and speed (e.g., C1200, D1200, and so on) to differentiate between classes of dialers (`centrex` or `DIMENSION PBX`). This is necessary because many larger offices may have more than one type of telephone network. One network may be dedicated to serving only internal office communications, while the other handles the external communications. Therefore, it is necessary to distinguish which line(s) should be used for internal communications and which should be used for external communications. The same distinction must be made in the `Systems` file because a match is made against the fourth field of `Systems` file entries as follows:

  ```
  Devices: ACU tty01,M - D1200 penril

  Systems: eagle Any ACU D1200 3-2-5-1 ogin: nuucp ssword: Oakgrass
  ```

  Some devices can be used at any speed, so the keyword `Any` may be used in the *Class* field. If `Any` is used, the line will match any speed requested in a `Systems` entry. If this field is `Any\` and the `Systems` *Class* field is `Any`, the speed defaults to 1200 bps.

- *Dialer-Token-Pairs*
  This field contains pairs of dialers and tokens. The dialer portion may be an automatic dial modem or direct for Direct Link devices. The token portion may be supplied immediately following the dialer; or if not present, it can be taken from the `Systems` file. This field has the following format:

  ```
  dialer-token dialer-token
  ```

where the last pair may or may not be present, depending on the associated device (dialer). In most cases, the last pair will contain only a dialer and the token is retrieved from the `Phone` field of the `Systems` entry. The *DTP* field may be structured in four ways, depending on the device associated with the entry:

1. If a direct link is established to a particular computer, the *DTP* field of the associated entry will contain the keyword `direct`. This is true for both types of direct link entries, `Direct` and *System-Name* (refer to discussion on the *Type* field).

2. If an automatic dialing modem is connected directly to a computer port, the *DTP* field of the associated `Devices` entry will only have one pair, normally the name of the modem. This name is used to match the particular `Devices` entry with an entry in the `Dialers` file. Therefore, this dialer must match the first field of a `Dialers` file entry as follows:

   ```
   Devices:  ACU acu1,M - 1200 ventel

   Dialers:  hayes2400 =&-% "" \M\r\p\r\c $ <K\T%%\r>\c ONLINE!\m
   ```

   Notice that only the dialer (`ventel`) is present in the *DTP* field of the `Devices` entry. This means that the token to be passed on to the dialer (in this case the telephone number) is taken from the *Phone* field of a `Systems` file entry.

3. If an automatic dialing modem is connected to a local area network (LAN), the computer must first access the switch and the switch will make the connection to the automatic dialing modem. This type of entry would have two pairs. The dialer portion of each pair (fifth and seventh fields of entry) is used to match entries in the `Dialers` file as follows:

   ```
   Devices:  ACU tty01 - 1200 develcon vent ventel

   Dialers:  ventel\f1 =&-% "" \M\r\p\r\c $ <K\T%%\r>\c ONLINE!\m

   Dialers:  develcon\f1 "" "" \pr\ps\c est:\077 \E\D\e \007
   ```

   In the first pair, `develcon` is the dialer and `vent` is the token that is passed to the Develcon switch to tell it which device (ventel modem) to connect to the computer. This token would be unique for each LAN switch since each switch may be set up differently. Once the ventel modem has been connected, the second pair is accessed where ventel is the dialer and the token is retrieved from the `Systems` file.

4. If a machine that you want to communicate with is on the same local network switch as your computer, your computer must first access the switch and then the switch can make the connection to the other machine. In this type of entry, there is only one pair. The dialer portion is used to match a `Dialers` entry as follows:

   Devices: `develcon tty01 - 1200 develcon \D`

   Dialers: `develcon "" "" \pr\ps\c est:\007 \E\D\e \007`

   As shown, the token is left blank to indicate that it is retrieved from the `Systems` file. The `Systems` file entry for this particular machine will contain the token in the *Phone* field that is normally reserved for the telephone number of the machine. This type of *DTP* contains an escape character (`\D`) that ensures that the contents of the *Phone* field will not be interpreted as a valid entry in the `Dialcodes` file.

There are two escape characters that may appear at the end of a *DTP* field:

`\T`	Indicates that the *Phone* (token) field should be translated using the `Dialcodes` file. This escape character is normally placed in the `Dialers` file for each caller script associated with an automatic dial modem (`penril`, `ventel`, etc.). Therefore, the translation will not take place until the caller script is accessed.
`\D`	Indicates that the *Phone* (token) field should not be translated using the `Dialcodes` file. If no escape character is specified at the end of a `Devices` entry, the `\D` is assumed (default). A `\D` is also used in the `Dialers` file with entries associated with network switches (`develcon` and `micom`).

11.8.2 `Dialers` *File*

The `Dialers` file (`/usr/lib/uucp/Dialers`) is used to specify the initial handshaking that must take place on a line before it can be made available for transferring data. This initial handshaking is usually a sequence of ASCII strings that are transmitted and expected and is often used to dial a telephone number using an ASCII dialer (such as the AT&T 2212C Modem). As shown in the above examples, the fifth field in a `Devices` file entry is used as an index into the `Dialers` file. Here an attempt is made to match the *Devices* field with the first field of each `Dialers` entry. In addition, each odd numbered *Devices* field starting with the seventh position is used as an index into the `Dialers` file. Changes must be made using one of the editors (`ed` or `vi`).

If the match succeeds, the `Dialers` entry is interpreted to perform the dialer negotiations. The first field matches the fifth and additional odd numbered fields in the `Devices` file. The second field is used as a translate string (the first of each pair of characters is mapped to the second character in the pair). This is usually used to translate = and – into whatever the dialer requires for "wait for dial tone" and "pause." The remaining fields are "expect-send" strings. The following `Dialers` file entries are typical examples:

```
att4000   =,-,  "" \M\dat\r\c OK\r \EATDT\T\r\c CONNECT \m\c
ventel    =&-%  "" \M\r\p\r\c $ <K\T%%\r>\c ONLINE!\m
hayes     =,-,  "" \M\dAT\r\c OK\r \EATDT\T\r\c CONNECT\m\c
vadic     =K-K  "" \005\p *-\005\p-*\005\p-* D\p BER? \E\T\ \r\c LINE
develcon        ""      "" \pr\ps\c est:\007 \E\D\ \007
micom     ""    ""\s\c NAME? \D\r\c GO
direct
hayes2400 =,-,  ""\M\pAT\r\dATQ0V1\v\c  OK\r  ATDT\T\r\m\c  CONNECT
```

The meaning of some of the escape characters (those beginning with \) used in the `Dialers` file are shown in the following list:

`\p`	Pauses (approximately ¼ to ½ second).
`\d`	Delays (approximately 2 seconds).
`\D`	Phone number or token without `Dialcodes` translation.
`\M`	Sets no modem control.
`\T`	Phone number or token with `Dialcodes` translation.
`\K`	Inserts a `BREAK`.
`\E`	Enables echo checking (for slow devices).

`\e`	Disables echo checking.
`\o`*digit(s)*	Sets timeout for send/expect.
`\r`	Carriage return.
`\c`	No new-line.
`\m`	Restores modem control.
`\n`	Sends new-line.
`\nnn`	Sends octal number.

Additional escape characters that may be used are listed in the section discussing the `Systems` file. The penril entry in the `Dialers` file is executed as follows. First, the telephone number argument is translated, replacing any = with a `W` (wait for dialtone) and replacing any - with a `P` (pause). The handshake given by the remainder of the line works as follows:

`""`	Waits for nothing.
`\d`	Delays for 2 seconds.
`>`	Waits for a >.
`s\p9\c`	Sends an `s`, pauses for ½ second, sends a `9`, sends no terminating new-line.
`)-W\p\r\ds\p9\c-)`	Waits for a `)`. If it is not received, processes the string between the - characters as follows. Sends a `W`, pauses, sends a carriage return, delays, sends an `s`, pauses, sends a `9` without a new-line and then waits for the `)`.
`y\c`	Sends a `y` without a new-line.
`:`	Waits for a `:`.
`\M`	Sets no modem control (CLOCAL).
`\m`	Restores modem control. Typically, CLOCAL is set for the duration of the dialer chat, then cleared (so `uucico`, `cu`, or `ct` will detect dropped lines) once connected to the remote system.
`\E\TP`	Enables echo checking. (From this point on, whenever a character is

	transmitted, it will wait for the character to be received before doing anything else.) Then, sends the telephone number followed by a pause character (`P`). The `\T` means take the telephone number passed as an argument and apply the `Dialcodes` translation and the modem function translation specified by field number 2 of this entry.
`>`	Waits for a >.
`9\c`	Sends a `9` without a new-line.
`OK`	Waits for the string `OK`.

11.8.3 `Systems` *File*

The `Systems` file (`/usr/lib/uucp/Systems`) contains the information needed by the `uucico` daemon to establish a communication link to a remote machine. Each entry in the file represents a machine that can be called by the computer. Furthermore, only those machines listed in the `Systems` file will be permitted to communicate with your computer via Basic Networking (UUCP) unless the execute permissions for `remote.unknown` are changed to permit communications with other machines. More than one entry may be present for a particular machine. The additional entries represent alternate communication paths that will be tried in sequential order.

Each entry in the `Systems` file has the following format:

System-Name Time Type Class Phone Login

Each field is defined in the following paragraphs.

- *System-Name*
  This field contains the node name of the remote machine.

- *Time*
  This field is a string that indicates the day of week and time of day when the remote machine can be called. The day portion may be a list containing some of the following:

  ```
  Su Mo Tu We Th Fr Sa
  ```

`Wk:`	For any weekday.

`Any:`	For any day.
`Never:`	For a passive arrangement with the remote machine. In this case, the computer will never initiate a call to the remote machine. The call must be initiated by the remote machine. The computer is in a passive mode in respect to the remote machine. (See the discussion of the `Permissions` file.)

The time should be a range of times such as 0800–1230. If no time portion is specified, any time of day is assumed to be allowed for the call. Note that a time range that spans 0000 is permitted. For example, 0800–0600 means all times are allowed other than times between 6 A.M. and 8 A.M. An optional subfield is available to specify the minimum time (in minutes) before a retry following a failed attempt. The subfield separator is a semicolon (;). For example, "\&`Any ;9`" is interpreted as call any time, but wait at least 9 minutes before retrying if a failure occurs.

- *Type*
  This field contains the device type that should be used to establish the communication link to the remote machine. The `Devices` file is searched for the device type listed, and the device found is used to establish the connection (if available). The following keywords may appear in this field:

`ACU`	This keyword indicates that the link to a remote computer is made through an automatic call unit (Automatic Dial Modem). This modem may be connected either directly to the computer or indirectly through a Local Area Network (LAN) switch.
Network	This keyword indicates that the link is established through a LAN switch, where `Network` is replaced with either `micom` or `develcon`. These two LAN switches are the only ones that contain caller scripts in the `Dialers` file. Other switches may be used if caller scripts are

	constructed and placed in the `Dialers` file.
System-Name	This keyword indicates a direct link to a particular machine where *System-Name* is replaced by the name of the particular computer (should be same as field one).

The keyword used in this field is matched against the first field of `Devices` file entries as follows:

```
Systems:  eagle Any ACU D1200 3-2-5-1 ogin: nuucp ssword: Oakgrass

Devices:  ACU tty01 - D1200 penril
```

- *Class*
  This field is used to indicate the transfer speed of the device used in establishing the communication link. It may contain a letter and speed (e.g., C1200, D1200, etc.) to differentiate between classes of dialers (refer to the discussion on the "Devices File," *Class* field). Some devices can be used at any speed, so the keyword `Any` may be used. This field must match the *Class* field in the associated `Devices` entry as follows:

```
Systems:  eagle Any ACU D1200 3-2-5-1 ogin: nuucp ssword: Oakgrass

Devices:  ACU tty01 - D1200 penril
```

- *Phone*
  This field is used to provide the telephone number (token) of the remote machine for automatic dialers (LAN switches). The telephone number is made up of an optional alphabetic abbreviation and a numeric part. The abbreviation must be one that is listed in the `Dialcodes` file. In this string, an equals sign (=) tells the ACU to wait for a secondary dial tone before dialing the remaining digits. A dash in the string (-) instructs the ACU to pause 4 seconds before dialing the next digit.

  If your computer is connected to a LAN switch, you may access other machines that are connected to that switch. The *Systems* entries for these machines will not have a telephone number in the *Phone* field. Instead, this field will contain the "token" that must be passed on to the switch so it will know which machine the computer wishes to communicate with. The associated `Devices` entry should have a `\D` at the end of the entry to ensure that this field is not translated using the `Dialcodes` file. For direct connections, the telephone field is ignored. A dash (-) should be used as a place holder.

- *Login*
  This field contains the login information given as a series of fields and subfields of the following format:

  [*expect send*] ...

  where *expect* is the string that is received and *send* is the string that is sent when the *expect* string is received. The *expect* field may be made up of subfields of the following form:

  expect[-*send*-*expect*]...

  where the *send* is sent if the prior *expect* is not successfully read and the *expect* following the *send* is the next expected string. For example, with `login--login`, UUCP will expect `login`. If UUCP gets `login`, it will go on to the next field. If it does not get `login`, it will send nothing followed by a new-line, then look for `login` again. If no characters are initially expected from the remote machine, the characters "" (null string) should be used in the first *expect* field. Note that all *send* fields will be sent followed by a new-line unless the *send* string is terminated with a `\c`.

There are several escape characters that cause specific actions when they are a part of a string sent during the login sequence. The following escape characters are useful in UUCP communications:

`\N`	Sends a null character.
`\b`	Sends a backspace character.
`\c`	If at the end of a string, suppresses the new-line that is normally sent. Ignored otherwise.
`\d`	Delays 2 seconds before sending or reading more characters.
`\p`	Pauses for approximately ¼ to ½ second.
`\n`	Sends a new-line character.
`\r`	Sends a carriage return.
`\s`	Sends a space character.
`\t`	Sends a tab character.

`\\`	Sends a \ character.
`\Onnn\`	Set timeout for expect-send.
`EOT`	Sends an EOT character (actually EOT new line is sent twice).
`BREAK`	Sends a break character.
`\ddd`	Collapses the octal digits (`ddd`) into a single character and sends that character.

11.8.4 `Dialcodes` *File*

The `Dialcodes` file (`/usr/lib/uucp/Dialcodes`) contains the dial-code abbreviations used in the *Phone* field of the `Systems` file. Each entry has the following format:

```
abb dial-seq
```

where `abb` is the abbreviation used in the `Systems` file (*Phone* field), and *dial-seq* is the dial sequence that is passed to the dialer when that particular `Systems` entry is accessed.

The entry

```
jt\ 9=847-
```

would be set up to work with a *Phone* field in the `Systems` file such as `jt7867`. When the entry containing `jt7867` is encountered, the sequence 9=847-7867 would be sent to the dialer.

11.8.5 `Permissions` *File*

The `Permissions` file (`/usr/lib/uucp/Permissions`) is used to specify the permissions that remote machines have with respect to login, file access, and command execution. Options are provided for restricting the ability to request files and the ability to receive files queued by the local site. In addition, an option is available to specify the commands that a remote site can execute on the local machine. Changes must be made using one of the editors (`vi` or `ed`).

11.8.5.1 How Entries Are Structured. Each entry is a logical line with physical lines terminated with a \ to indicate continuation. Entries are made up of options delimited by white space. Each option is a name/value pair. These are constructed by an option name followed by an = and the value. Note that no white space is allowed within an option assignment.

Comment lines begin with a pound sign (#), and they occupy the entire line up to a new-line character. Blank lines are ignored (even within multi-line entries). There are two types of `Permissions` entries:

`LOGNAME`	Specifies permissions that take effect when a remote machine logs in on (calls) your computer.
`MACHINE`	Specifies permissions that take effect when your computer logs in on (calls) a remote machine.

`LOGNAME` entries contain a `LOGNAME` option, and `MACHINE` entries contain a `MACHINE` option.

11.8.5.2 Considerations. The following items should be considered when using the `Permissions` file to restrict the level of access granted to remote machines:

1. All login IDs used by remote machines to log in for UUCP-type communications must appear in one and only one `LOGNAME` entry.
2. Any site that is called whose name does not appear in a `MACHINE` entry will have the following default permissions/restrictions:
   - Local send and receive requests will be executed.
   - The remote machine can send files to your computer `/usr/spool/uucppublic` directory.
   - The commands sent by remote machine for execution on your computer must be one of the default commands, usually `rmail`.

11.8.5.3 Options. This section provides the details of each option, specifying how they are used and their default values.

- `REQUEST`
  When a remote machine calls your computer and requests to receive a file, this request can be granted or denied. The `REQUEST` option specifies whether or not the remote machine can request to set up file transfers from your computer. The string:

```
REQUEST=yes
```

specifies that the remote machine can request to transfer files from your computer. The string:

```
REQUEST=no
```

specifies that the remote machine cannot request to receive files from your computer. The no string is the default value. It will be used if the REQUEST option is not specified. The REQUEST option can appear in either a LOGNAME (remote calls you) entry or a MACHINE (you call remote) entry.

- SENDFILES
  When a remote machine calls your computer and completes its work, it may attempt to take work that your computer has queued for it. The SENDFILES option specifies whether or not your computer can send the work queued for the remote machine. The string:

  ```
  SENDFILES=yes
  ```

  specifies that the computer may send the work that is queued for the remote machine as long as it logged in as one of the names in the LOGNAME option. This string is mandatory if the computer is in a "passive mode" with respect to the remote machine. The string:

  ```
  SENDFILES=call
  ```

  specifies that files queued in your computer will only be sent when the computer calls the remote machine. The call value is the default for the SENDFILE option. This option is only significant in LOGNAME entries since MACHINE entries apply when calls are made out to remote machines. If the option is used with a MACHINE entry, it will be ignored.

- READ and WRITE
  These options specify the various parts of the file system that uucico can read from or write to. The READ and WRITE options can be used with either MACHINE or LOGNAME entries.

  The default for both the READ and WRITE options is the uucppublic directory, as shown in the following strings:

  ```
  READ=/usr/spool/uucppublic WRITE=/usr/spool/uucppublic
  ```

  The strings:

  ```
  READ=/ WRITE=/
  ```

specify permission to access any file that can be accessed by a local user with "other" permissions.

The value of these entries is a list of path names separated by colons. The READ option is for requesting files, and the WRITE option is for depositing files. One of the values must be the prefix of any full path name of a file coming in or going out. To grant permission to deposit files in /usr/news as well as the public directory, the following values should be used with the WRITE option:

```
WRITE=/usr/spool/uucppublic:/usr/news
```

It should be pointed out that if the READ and WRITE options are used, all path names must be specified because the path names are not added to the default list. For instance, if the /usr/news path name was the only one specified in a WRITE option, permission to deposit files in the public directory would be denied.

- NOREAD and NOWRITE
  The NOREAD and NOWRITE options specify exceptions to the READ and WRITE options or defaults. The strings:

  ```
  READ=/ NOREAD=/etc WRITE=/usr/spool/uucppublic
  ```

  would permit reading any file except those in the /etc directory (and its subdirectories – remember, these are prefixes) and writing only to the default /usr/spool/uucppublic directory. NOWRITE works in the same manner as the NOREAD option. NOREAD and NOWRITE can be used in both LOGNAME and MACHINE entries.

- CALLBACK
  The CALLBACK option is used in LOGNAME entries to specify that no transaction will take place until the calling system is called back. The string:

  ```
  CALLBACK=yes
  ```

  specifies that your computer must call the remote machine back before any file transfers will take place.

  The default for the CALLBACK option is:

  ```
  CALLBACK=no
  ```

The CALLBACK option is very rarely used. Note that if two sites have this option set to yes for each other, a conversation will never get started.

- COMMANDS

☛ The COMMANDS option can be hazardous to the security of your system. Use it with extreme care.

The uux program will generate remote execution requests and queue them to be transferred to the remote machine. Files and a command are sent to the target machine for remote execution. The COMMANDS option can be used in MACHINE entries to specify the commands that a remote machine can execute on your computer. The string:

```
COMMANDS=rmail
```

indicates the default commands that a remote machine can execute on your computer. If a command string is used in a MACHINE entry, the default commands will be overridden. For instance, the entry:

```
MACHINE=owl:raven:hawk:dove \
COMMANDS=rmail:rnews:lp
```

overrides the COMMAND default such that the command list for machines owl, raven, hawk, and dove now consists of rmail, rnews, and lp. In addition to the names as specified above, there can be full path names of commands. For example:

```
COMMANDS=rmail:/usr/lbin/rnews:/usr/local/lp
```

specifies that command rmail uses the default path. The default paths for the computer are /bin, /usr/bin, and /usr/lbin. When the remote machine specifies rnews or /usr/lbin/rnews for the command to be executed, /usr/lbin/rnews will be executed regardless of the default path. Likewise, /usr/local/lp is the lp command that will be executed.

Including the ALL value in the list means that any command from the remote machine(s) specified in the entry will be executed. If you use this value, you give the remote machine full access to your computer.

The string:

```
COMMANDS=/usr/lbin/rnews:ALL:/usr/local/lp
```

illustrates two points. The ALL value can appear anywhere in the string, and the path names specified for rnews and lp will be used (instead of the default) if the requested command does not contain the full path names for rnews or lp.

The VALIDATE option should be used with the COMMANDS option whenever potentially dangerous commands like cat and uucp are specified with the COMMANDS option. Any command that reads or writes files is potentially dangerous to local security when executed by the UUCP remote execution daemon (uuxqt).

- VALIDATE
  The VALIDATE option is used with the COMMANDS option when specifying potentially dangerous commands. It is used to provide a certain degree of verification of the caller's identity. The use of the VALIDATE option requires that privileged machines have a unique login/password for UUCP transactions. An important aspect of this validation is that the login/password associated with this entry be protected. If an outsider gets that information, that particular VALIDATE option can no longer be considered secure.

  A great deal of consideration should be given to providing a remote machine with a privileged login and password for UUCP transactions. Giving a remote machine a special login and password with file access and remote execution capability is like giving anyone on that machine a normal login and password on your computer. Therefore, if you cannot trust someone on the remote machine, do not provide that machine with a privileged login and password.

  The LOGNAME entry:

  ```
  LOGNAME=uucpfriend VALIDATE=eagle:owl:hawk
  ```

  specifies that if one of the remote machines that claims to be eagle, owl, or hawk logs in on your computer, it must have used the login uucpfriend. As can be seen, if an outsider gets the uucpfriend login/password, masquerading is trivial. But what does this have to do with the COMMANDS option that only appears in MACHINE entries? It links the MACHINE entry (and COMMANDS option) with a LOGNAME entry associated with a privileged login. This link is needed because the execution daemon is not running while the remote machine is logged in. In fact, it is an asynchronous process with no knowledge of what machine sent the execution request. Therefore, the real question is how does your computer know where the execution files came from?

Each remote machine has its own "spool" directory on your computer. These spool directories have write permission given only to the UUCP programs. The execution files from the remote machine are put in its spool directory after being transferred to your computer. When the uuxqt daemon runs, it can use the spool directory name to find the MACHINE entry in the Permissions file and get the COMMANDS list, or if the machine name does not appear in the Permissions file, the default list will be used.

The following example shows the relationship between the MACHINE and LOGNAME entries:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
COMMANDS=ALL \
READ=/  WRITE=/

LOGNAME=uucpz VALIDATE=eagle:owl:hawk \
REQUEST=yes SENDFILES=yes \
READ=/  WRITE=/
```

These entries provide unlimited read, write, and command execution for the remote machines eagle, owl, and hawk. The ALL value in the COMMANDS option means that any command can be executed by either of these machines. Using the ALL value gives the remote machine unlimited access to your computer. In fact, files that are only readable or writable by user uucp (like Systems or Devices) can be accessed using commands like ed. This means a user on one of the privileged machines can write in the Systems file as well as read it.

In the first entry, you must make the assumption that when you want to call one of the machines listed, you are really calling either eagle, owl, or hawk. Therefore, any files put into one of the eagle, owl, or hawk spool directories are put there by one of those machines. If a remote machine logs in and says that it is one of these three machines, its execution files will also be put in the privileged spool directory. You, therefore, have to validate that the machine has the privileged login uucpz.

11.8.5.4 `MACHINE` *Entry for "Other" Systems.* You may want to specify different option values for the machines your computer calls that are not mentioned in specific `MACHINE` entries. This may occur when there are many machines calling in, and the command set changes from time to time. The name `OTHER` for the machine name is used for this entry as follows:

```
MACHINE=OTHER \
COMMANDS=rmail:rnews:/usr/lbin/Photo:/usr/lbin/xp
```

All other options available for the `MACHINE` entry may also be set for the machines that are not mentioned in other `MACHINE` entries.

11.8.5.5 Combining `MACHINE` *and* `LOGNAME` *Entries.* It is possible to combine `MACHINE` and `LOGNAME` entries into a single entry where the common options are the same. For example, the two entries:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
  READ=/  WRITE=/

LOGNAME=uucpz REQUEST=yes SENDFILES=yes \
  READ=/  WRITE=/
```

share the same `REQUEST`, `READ`, and `WRITE` options. These two entries can be merged into one entry as follows:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
LOGNAME=uucpz SENDFILES=yes \
    READ=/  WRITE=/
```

11.8.5.6 Sample `Permissions` *Files.*

1. *Example 1*

   This first example represents the most restrictive access to your computer.

   ```
   LOGNAME=nuucp
   ```

   It states that login `nuucp` has all the default permissions/restrictions:

   - The remote machine can only send files to `uucppublic`.
   - The remote machine cannot request to receive files (`REQUEST` option).
   - No files that are queued for the remote machine will be transferred during the current session (`SENDFILES` option).

- The only commands that can be executed are the defaults.

This entry alone is sufficient to start communications with remote machines, permitting files to be transferred only to the `/usr/spool/uucppublic` directory.

2. *Example 2*

The next example is for remote machines that log in but have fewer restrictions. The login and password corresponding to this entry should not be distributed to the general public; it is usually reserved for closely coupled systems where the `Systems` file information can be tightly controlled.

```
LOGNAME=uucpz REQUEST=yes SENDFILES=yes \
  READ=/  WRITE=/
```

This entry places the following permissions/restrictions on a machine that logs in as `uucpz`:

- Files can be requested from your computer (`REQUEST` option).
- Files can be transferred to any directory or any file that is writable by user "other," that is, a file/directory that is writable by a local user with neither owner nor group permissions (`WRITE` option).
- Any files readable by user "other" can be requested (`READ` option).
- Any requests queued for the remote machine will be executed during the current session. These are files destined for the machine that has called in (`SENDFILES` option).
- The commands sent for execution on the local machine must be in the default set.

3. *Example 3*

The two previous examples showed entries that referred to remote machines when they log in to your computer. This example is an entry used when calling remote machines:

```
MACHINE=eagle:owl:hawk:raven   REQUEST=yes READ=/  WRITE=/
```

When calling any of the systems given in the `MACHINE` list, the following permissions prevail:

- The remote machine can both request and send files (REQUEST option).
- The source or destination of the files on the local machine can be anywhere in the file system (with read/write option).
- The only commands that will be executed for the remote machine are those in the default set.

Any site that is called that does not have its name in a MACHINE entry will have the default permissions as stated in Example 1, with the exception that files queued for that machine will be sent. (The SENDFILES option is only interpreted in the LOGNAME entry.)

11.8.6 Poll File

The Poll file (/usr/lib/uucp/Poll) contains information for polling specified machines. Each entry in the Poll file contains the name of the remote machine to call, followed by a TAB character, and finally the hours the machine should be called. The entry:

```
eagle   0 4 8 12 16 20
```

will provide polling of machine eagle every 4 hours.

☛ It should be understood that uudemon.poll does not actually perform the poll, it merely sets up a polling work (C.) file in the spool directory that will be seen by the scheduler, started by uudemon.hour. Refer to the discussion on uudemon.poll.

11.8.7 Maxuuxqts File

The Maxuuxqts (/usr/lib/uucp/Maxuuxqts) file contains an ASCII number to limit the number of simultaneous uuxqt programs running. This file is delivered with a default entry of 2. This may be changed to meet local needs. If there is a lot of traffic from mail, it may be advisable to increase the number of uuxqt programs that will run to reduce the time it takes for the mail to leave your system. However, keep in mind that the load on the system increases with the number of uuxqt programs running.

11.8.8 Maxuuscheds File

The Maxuuscheds (/usr/lib/uucp/Maxuuscheds) file contains an ASCII number to limit the number of simultaneous uusched programs running. Each uusched running will have one uucico associated with it; limiting the number will directly affect the load on the system. The limit should be less than the number of outgoing lines used by UUCP (a smaller number is often desirable). This file is delivered with a default entry of 2. Again, this may be changed to meet the needs of the local system. However, keep in mind that the load on the system increases with the number of uusched programs running.

11.8.9 The remote.unknown *Program*

/usr/lib/uucp/remote.unknown, which is the remote.unknown program, is a shell file that is executed when a remote site that is not in the Systems file calls in to start a conversation. The shell script will append the name and time information to the file /usr/spool/uucp/.Admin/Foreign. Since it is a shell, it can be easily modified. For example, it can be set up to send mail to the administrator. The contents of this file, as delivered, is as follows:

```
FOREIGN=/usr/spool/uucp/.Admin/Foreign
echo "`date`: call from system $1" >>$FOREIGN
```

If you want to permit machines that are not listed in your Systems file to communicate via Basic Networking, remove the execute permissions from the remote.unknown file. For example:

```
chmod 444 /usr/lib/uucp/remote.unknown
```

When remote.unknown is executable, your computer will hang up if a machine that is not in your Systems file calls in (to UUCP) on your system.

11.9 Administrative Tasks

There is a minimum amount of maintenance that must be applied to your computer to keep the files updated, to ensure that the network is running properly, and to track down line problems. When more than one remote machine is involved, the job becomes more difficult because there are more files to update and because users are much less patient when failures occur between machines that are under local control. The uustat program provides you with information about the latest attempts to contact various machines and the age and number of jobs in the queue for remote machines. The following sections describe the routine administrative tasks that must be performed by someone acting as the UUCP administrator or are automatically performed by the UUCP daemons.

The biggest problem in a dialup network like UUCP is dealing with the backlog of jobs that cannot be transmitted to other machines. The following cleanup activities should be routinely performed.

11.9.1 Cleanup of Undeliverable Jobs

The `uustat` program should be invoked regularly to provide information about the status of connections to various machines and the size and age of the queued requests. The `uudemon.admin` shell should be started by `cron` at least once per day. This will send the administrator the current status. Of particular interest is the age (in days) of the oldest request in each queue, the number of times a failure has occurred when attempting to reach that machine, and the reason for failure. In addition, the age of the oldest execution request (`X.` file) is also given.

The `uudemon.cleanu` shell file is set up to remove any jobs that have been queued for several days and cannot be sent. Left-over data (`D.`) and work (`C.`) files are removed after 7 days, and execute (`X.`) files are removed after 2 days. It also provides feedback to the user indicating when jobs are not being accomplished and when these jobs are being deleted.

11.9.2 Cleanup of the Public Area

To keep the local file system from overflowing when files are sent to the public area, the `uudemon.cleanu` procedure is set up with a `find` command to remove any files that are older than 7 days and directories that are empty. This interval may need to be shortened by changing the `uudemon.cleanu` shell file if there is not sufficient space to devote to the public area.

Since the spool directory is very dynamic, it may grow large before transfers take place. Therefore, it is a good idea to reorganize its structure. The best way to do this on your computer is to use the `crontab` command to clean out the spool directory at a specified time.

First, specify the file you want to have the cleanup code in as follows:

```
crontab clean.wk
```

The *clean.wk* file will contain the code for all files cleaned at a specified time (every Monday, for example), based on the time specified in the `crontab` file. You may already have entries in *clean.wk* which means you will also have the cleanup time specified. See *crontab*(1) for additional information. If you wish to specify a new cleanup time, first, make a new file with the `crontab` command as above. Edit the `crontab` file to specify the time of cleanup. For example:

```
0 0 1 15 * 1
```

in the `crontab` file would indicate cleanup on the first and fifteenth of each month, as well as on every Monday. In the file you specified with the `crontab` command, enter the following code (the # sign lines are comment lines):

```
#       Clean up /usr/spool/uucp
#       Most cleanup is now done by uudemon.cleanu
#       so just copy out and back.
#
echo "UUCP SPOOL DIRECTORIES CLEANUP STARTED"
#
cd /usr/spool/uucp
mkdir ../nuucp
chown uucp ../nuucp
chgrp uucp ../nuucp
find . -print|cpio -pdml ../nuucp
cd ..
mv uucp ouucp
mv nuucp uucp
rm -rf ouucp
rm -f /usr/spool/locks/LCK*
#
#       Note:
#       Change the tty?? device to the
#       device you are using for UUCP.
#       For example change tty?? to tty01.
#
chown uucp /dev/tty??
chgrp uucp /dev/tty??
chmod 0644 /dev/tty??
chmod 0222 /dev/tty??
echo "UUCP SPOOL DIRECTORIES CLEANUP FINISHED"
```

11.9.3 Compaction of Log Files

This version of Basic Networking has individual log files for each machine and each program. For example, machine `eagle` has a log file for `uucico` requests and a log file for `uuxqt` execution requests. The `uulog` program gives the user access to the information in these files by machine name. These files are combined and stored in directory `/usr/lib/uucp/.Old` whenever `uudemon.cleanu` is executed. This shell script saves files that are 2 days old. The 2 days can be easily changed by changing the appropriate line in the `uudemon.cleanu` shell. If space is a problem, the administrator might consider reducing the number of days the files are kept.

11.9.4 Cleanup of `sulog` *and* `cron/log`

The `/usr/adm/sulog` and `/usr/lib/cron/log` files are both indirectly related to UUCP transactions. The `sulog` file contains a history of the `su` command usage. Since each `uudemon` entry in the `/usr/spool/cron/crontab/root` file uses the `su` command, the `sulog` could become rather large over a period of time. The `sulog` should be purged periodically to keep the file at a reasonable size.

Similarly, a history of all processes spawned by `/etc/cron` are recorded in `/usr/lib/cron/log`. The `cron/log` file will also become large over a period of time and should be purged periodically to limit its size.

11.10 UUCP and `cron`

The `cron` daemon is a tool that proves to be very useful in the administration of UNIX Systems. When the computer is in run state 2 (multi-user), the `cron` daemon scans the `/usr/spool/cron/crontab/root` file every minute for entries that contain "work" scheduled to be executed at that time. It is recommended that the UUCP administrator make use of `cron` to aid in the administration of Basic Networking.

As delivered, Basic Networking contains four entries in the `root` `crontab` file. Each one of these entries executes shell scripts that are used for various administrative purposes. These shell scripts can be easily modified to meet the needs of your system.

11.10.1 `uudemon.admin`

The `uudemon.admin` shell script mails status information to the UUCP administrative login (`uucp`) using `uustat` commands with the `-p` and `-q` options. Refer to *uustat*(1C) for interpretation of these options.

The `uudemon.admin` shell script should be executed daily by an entry in the `root` `crontab` file. The default `root` `crontab` entry for `uudemon.admin` is as follows:

```
48 11,14, ** 1-5 /bin/su uucp -c "/usr/lib/uucp/uudemon.admin" >

/dev/null 2>&1
```

11.10.2 uudemon.cleanu

The `uudemon.cleanu` shell script cleans up the Basic Networking log files and directories. Archived log files are updated so that no log information over 3 days old is kept. Log files for individual machines are taken from the `/usr/spool/uucp/.Log` directory, merged, and placed in the `/usr/spool/uucp/.Old` directory along with the older log information. Files and directories that are no longer needed in the spool directories are removed. After cleanup is performed, the UUCP administrative login (`uucp`) is mailed a summary of the status information gathered during the current day.

The `uudemon.cleanu` shell script should be executed by an entry in the `root crontab` file. It can be run daily, weekly, or whenever, depending on the amount of UUCP traffic that enters and leaves your computer. The default `root crontab` entry for `uudemon.cleanu` is as follows:

```
45 23 * * * ulimit 5000; /bin/su uucp -c

"/usr/lib/uucp/uudemon.cleanu" > /dev/null 2>&1
```

If log files get very large, the `ulimit` may need to be increased.

11.10.3 uudemon.hour

The `uudemon.hour` shell script is used to call UUCP programs on an hourly basis. The `uusched` program is called to search the spool directory for work files (C.) that have not been processed and schedule these files for transfer to a remote machine. The `uuxqt` daemon is called to search the spool directory for execute files (*X*/C.) that have been transferred to your computer and were not processed at the time they were transferred.

The `uudemon.hour` shell script should be executed by an entry in the `root crontab` file. If the amount of traffic leaving and entering your computer is large, it may be started once or twice an hour. If it is small, it may be started once every 4 hours or so. The default `root crontab` entry for `uudemon.hour` is as follows:

```
26,56 * * * * /bin/su uucp -c

"/usr/lib/uucp/uudemon.hour" > /dev/null 2>&1
```

11.10.4 uudemon.poll

The uudemon.poll shell script is used to poll the remote machines listed in the Poll file (/usr/lib/uucp/Poll). It creates work files (C.) for machines according to the entries listed in the Poll file. It should be set up to run once an hour just prior to uudemon.hour so that the work files will be present when uudemon.hour is called.

The uudemon.poll script should be executed by an entry in the root crontab file. The exact times it runs is dependent on the scheduling of uudemon.hour. The default root crontab entry for uudemon.poll is as follows:

```
40 * * * * /bin/su uucp -c "/usr/lib/uucp/uudemon.poll" >

/dev/null 2>&1
```

Notice how uudemon.poll is scheduled to run 11 minutes before uudemon.hour runs.

11.11 UUCP Logins and Passwords

There are two login IDs associated with Basic Networking: one is the UUCP and the other is an access login, nuucp, used by remote computers to access your computer. These logins should not be changed from their default settings of uucp and nuucp.

The uucp administrative login is the owner of all the UUCP object and spooled data files. The following is a sample entry in the /etc/passwd file for the administrative login:

```
uucp:x:5:1:UUCP.Admin:/usr/lib/uucp:
```

The nuucp access login allows remote machines to log in on your computer. The following is a sample entry in the /etc/passwd file for the access login:

```
nuucp:x:6:1:UUCP.Admin:/usr/spool/uucppublic:

/usr/lib/uucp/uucico
```

Notice that the standard shell is not given to the nuucp login. The shell that nuucp receives is the uucico daemon that controls the conversation when a remote machine logs in to your machine.

The assigning of passwords for the uucp and nuucp logins is left up to the administrator. The passwords should be at least six to eight characters. Only the first eight characters of the passwords are significant. If the password for the access login is changed for security reasons, make certain that the remote machines that are a part of your network are properly notified of the change.

12. ELECTRONIC MAIL OPTIONS

Electronic mail is a form of communication used on computers to send messages to and receive messages from other users. You can exchange mail locally with other users on your computer, with users on other computers within your company over a network, or with users almost anywhere in the world, if your computer is tied in to one of the international networks.

Electronic mail has two major components:

The user interface
The user interface is the part of the mail system with which the user interacts to send and receive mail. It picks up the mail in a known location and transfers it to the user's mailbox.

The mail transport
The mail transport is that part of the mail system which puts the mail received in a known location and sends the mail given by the user interface program to the desired destination.

Users interact directly with the user interface program and in most cases are not even aware of the existence and function of the mail transport program.

12.1 The User Interface

Two different user interfaces for the mail system are available with the INTERACTIVE UNIX Operating System. One is a simple "character command" interface called `mailx`; the other is a "menu driven" interface called the TEN/PLUS* Mail System.

12.2 The Mail Transport Program (`sendmail`)

The mail transport program used in the INTERACTIVE UNIX System is `sendmail`. Refer to "Sendmail – An Internetwork Mail Router" in this guide for more information.

12.3 Installation

Before mail can be exchanged, the following tasks must be performed:

- Establish a physical connection between the various computers. These connections can be established using modems or direct connections.

- Assign system names to your computers.

- Assign a password to the `nuucp` login on each computer for other computers to use when sending mail. This will ensure that only friendly computers can log into your system (when using `uucp`).
- Set up the `uucp` program, if you plan to use it, or install and set up TCP/IP if you plan to use that transport protocol with `sendmail`.
- Provide your system name, login and password, data telephone number, and communication attributes to the system administrators of computers with which you will be exchanging mail.
- Obtain the system names, logins and passwords, telephone numbers, and attributes for their computers from the other system administrators.

When you install the Basic Networking optional subset on versions 2.2 and later of the INTERACTIVE UNIX Operating System, the `sendmail` mail transport program and the character-based user interface, `mailx`, are installed. You should then configure the `sendmail` mail transport using the `sysadm mailmgmt` command. Refer to the "Sendmail Installation and Operation Guide" for information about configuring `sendmail`. Refer to *mailx*(1) for more information about using that user interface to mail.

To use the TEN/PLUS Mail System as the user interface for electronic mail, you must install the TEN/PLUS User Interface and TEN/PLUS Mail System optional subset. Please refer to the "TEN/PLUS Mail System Tutorial" and the "TEN/PLUS Mail System Reference Manual" in the *TEN/PLUS Mail System Guide* for more information about using this menu-driven interface.

13. TROUBLESHOOTING

This section describes some of the most frequently asked questions that arise during the installation and configuration of the INTERACTIVE UNIX Operating System. These may help you to troubleshoot problems with your system.

- **What does the error message `Interrupt conflict between wt and gendev` mean when building a kernel?**

  Generally this means that there is a conflict between two (or more) drivers in their interrupt level. The most common cause for this is that you have not used the `kconfig` utility to previously configure the High Performance Device Driver. The default HPDD is delivered with four interrupts configured (5, 11, 14, 15). If you configure the HPDD, you will generally free up the interrupts needed by other devices.

  Refer to the driver conflict information found in section 8, "HARDWARE COMPATIBILITY AND CONFIGURATION," for the appropriate drivers for your devices.

- **What does the error message `getty respawning too rapidly` mean?**

  There are two common causes for this, both related to the asynchronous (`asy`) driver.

  - 1a) A serial terminal may be configured with the incorrect baud rate or it may be improperly wired. Refer to the documentation that accompanied your hardware to verify the correct baud rate and proper wiring. 1b) The second `asy` port may not be configured properly into the kernel. Refer to section 8.4, "The Asynchronous Port Driver," to verify the proper configuration of the second `asy` device and use the `kconfig` program to verify that the second `asy` device is configured. To do this, select `Drivers` under `Configure` on the bar menu. Select the `asy` driver and access the `Driver Configuration Data` form. Select `asy2` from the list of `asy` drivers to review its configuration. If the `Driver Status` button has an asterisk in the `On` field, then it is configured into the current kernel.

  - 2) A modem may be configured incorrectly, or you may be using a device name that does not support modem control.

The modem should be in "quiet" mode, that is, its echo and verbose modes should be OFF. The modem should use a device that uses modem control, such as `/dev/ttyd0` or `/dev/ttyd3`, rather than `/dev/tty00` or `/dev/tty03`, which do not.

- **My printer only gives me blank pages.**

  This is most probably caused by the printer not interpreting the NL character that the INTERACTIVE UNIX System provides for new lines. The DOS environment uses a combination of carriage return and line feed (CR/LF). This problem can be solved by using the INTERACTIVE UNIX System `utod` program, which translates the UNIX System NL characters to CR/LF and passes command sequences without translation. Replace the word `cat` in the file in `/usr/spool/lp/model/*` (your printer mode) with `utod`.

- **My printer does not print at all.**

  First, check the physical connections to the printer. Then, to determine the proper INTERACTIVE UNIX System device for the printer, type:

  ```
  cat /dev/passwd > /dev/lp1
  ```

  If this is the correct device, a copy of `/etc/passwd` will print on your printer. Although `/dev/lp1` is correct on most machines, you may have to try `/dev/lp0` or `/dev/lp2` if `/dev/lp1` does not work. When you have determined the proper device, reinstall your printer using the `sysadm lpmgmt` menus.

- **How do I change my `ULIMIT`?**

  `ULIMIT` may be changed in two places. First, to change the `ULIMIT` for users logging in from the console, virtual terminals, or serial terminals, edit the file `/etc/default/login` and change the `ULIMIT` entry to the desired number. Remember that the entry is in 512-byte blocks, so an entry of 4096 indicates that the largest file size will be about 2 MB. To change the `ULIMIT` for users logging in over Ethernet, the `ULIMIT` tunable parameter must be changed. Access the `kconfig` program and select `Kernel` on the `Parameters` menu under `Configure` on the bar menu. When asked for the value, change the default value that is displayed to the value desired (remembering that 1024 = approximately 1 MB). After you have rebuilt the kernel and rebooted the system, the new ULIMIT will be in effect for Ethernet login.

☛ It is not necessary to rebuild the kernel to adjust the ULIMIT for non-Ethernet logins.

- **How do I verify that my mouse is working?**

  The easiest way to do this is to type:

  ```
  cat < /dev/tty00
  ```

  for a serial mouse using /dev/tty00, or to type:

  ```
  cat < /dev/mouse
  ```

  for a bus mouse using /dev/mouse. The bus mouse simply returns the prompt. If you have a serial mouse, move the mouse and see if something (other than ASCII characters) appears on the screen. If the prompt does not return (bus mouse) or nothing appears (serial mouse), check your configuration against the appropriate mouse driver configuration in section 8, "HARDWARE COMPATIBILITY AND CONFIGURATION."

- **I keep building kernels but no change occurs.**

  Check to be sure that there is no kernel (a file named unix) in the /etc/conf/cf.d directory. Kernels present in this directory are automatically linked to /unix at shutdown time.

- **How do I mount the DOS partition of my fixed disk or a DOS diskette?**

  Use the following command for a partition:

  ```
  mount -f DOS /dev/dsk/0p? /mnt
  ```

  Replace the ? with the number of the fdisk partition on which DOS resides. Use the following command for a diskette:

  ```
  mount -f DOS /dev/dsk/f0q15dt /mnt
  ```

  Note that this device name mounts a 1.2 MB 5 ¼ inch diskette.

- **I modified the /etc/inittab file and built a new kernel but my changes did not remain.**

  You must change both /etc/conf/node.d/*filename* and /etc/conf/cf.d/init.base to make permanent changes in the inittab file.

- **I can't install my system. All I get is** `booting the UNIX system` **and then my disk light goes out.**

  This may be caused by hardware that is not being recognized correctly. Simplify the hardware environment by removing unnecessary adapters and verify that the INTERACTIVE UNIX Operating System is supported on your hardware. This problem may also be caused by bad media. If your *Boot* and *Install* diskettes cannot be used on another system, contact your distributor for a replacement.

- **I've built a kernel that won't boot and I can't find** `OLD.unix` **to boot an old kernel. Now what can I do?**

  This problem can be solved by mounting the fixed disk partition on the boot disk and copying the kernel found on the *Boot* diskette to the INTERACTIVE UNIX System partition on the fixed disk. Do the following:

  1. Insert the *Boot* diskette into the diskette drive and press $\boxed{\text{RESET}}$.

  2. Remove the *Boot* diskette and insert the *Install* diskette into the diskette drive when prompted to do so.

  3. At this point, you may or may not be asked to authenticate your software. If you are asked, type in the appropriate serial number and authorization code. In any case, *after* the system asks you to choose your keyboard nationality, press $\boxed{\text{CTRL}}$ $\boxed{\backslash}$ to break out of the installation procedure.

  4. Run `fsck` on the `root` file system on your fixed disk (from the `root` prompt) by typing:

     ```
     # /etc/fsck /dev/dsk/0s1
     ```

     The `fsck` command will either run with no errors or request action from the user on repairing the file system. Most of the time answering "yes" to the questions ask by `fsck` will be sufficient, but be aware this could remove some files.

  5. Mount the fixed disk (device `0s1`) by typing:

     ```
     # mount /dev/dsk/0s1 /mnt
     ```

6. Type:

   ```
   # init switchroot
   ```

   to change `root` to the fixed disk.

7. Type:

   ```
   # exit
   ```

   and ignore any messages that appear at this point.

8. Remove the *Install* diskette and insert the *Boot* diskette.

9. Mount the diskette by typing:

   ```
   # mount /dev/dsk/f0q15d /mnt
   ```

10. Copy the kernel (`/unix`) from the *Boot* diskette to the fixed disk by typing:

    ```
    # cp /mnt/unix /unix
    ```

11. Unmount the *Boot* diskette by typing:

    ```
    # umount /mnt
    ```

12. Remove the *Boot* diskette and type:

    ```
    # sync; sync; uadmin 2 0
    ```

    to shut down your system cleanly. Press any key to reboot when prompted.

This will place a bootable kernel in your INTERACTIVE UNIX System partition. At this point your fixed disk should boot and behave normally.

☛ When you get the INTERACTIVE UNIX System prompt, make sure you are logged in as `root`. You should then reinstall all drivers previously installed using the `kconfig` command. Refer to *kconfig*(1) for more information.

Appendix: KERNEL ERROR MESSAGES

1. INTRODUCTION

The INTERACTIVE UNIX System kernel displays an error message on the system console when it encounters an internal error condition. If you are using virtual terminals, you will not see an error message unless you are switched to the console virtual terminal.

Kernel error messages are divided into three severity classes: PANIC, WARNING, and NOTICE. The severity class of a message is displayed at the beginning of the message.

2. PANIC MESSAGES

The kernel displays a PANIC error message when it encounters a problem that is so severe that it cannot continue to function and must stop. This is usually caused by a hardware problem, an incorrectly configured device, or a logic error in a device driver that has been configured into the kernel. It is only occasionally caused by a logic error in the kernel itself.

The following are the most common PANIC error messages:

```
Kernel mode trap. Type 0xD (general protection exception)

Kernel mode trap. Type 0xE (page fault)

User mode trap. Type 0xD

User mode trap. Type 0xE
```

PANIC error messages can have many causes. If your system displays a PANIC message after having run without problems for some time, it is likely that there has simply been a random hardware error that will not happen again soon. Turn off the power, wait 30 seconds, and turn the power back on. The system should check the disk file systems, correct any problems it finds, and resume normal operation. If the system resumes operation without problems and does not produce the error message again when you again do whatever you were doing on the system when it first appeared, then your system is probably fine. On the other hand, if the previous error message or some other PANIC error message appears again, then your system has a problem that must be corrected.

You can look in `/usr/include/sys/trap.h` for a description of trap-type values if you get the message `Kernel mode trap. Type 0xD (general protection exception).`

If you have reconfigured the kernel recently, you probably have a configuration or driver problem. If you have not reconfigured the kernel recently, then you probably have a hardware problem. If you have added a device driver supplied by someone other than INTERACTIVE, there may be an error in the driver software logic.

The following messages indicate a hardware parity error:

```
Parity error address unknown
```

```
Parity error at address 0x
```

```
FATAL: Parity error on an add-on card
```

The most common causes are:

- RAM memory failure has occurred. This may be a random error that will not occur again for a long time, or it may have been caused by a permanent memory chip failure. Turn the power off and on to reboot the system, and if the problem recurs, have the hardware repaired.
- A device configuration conflict has occurred. This will occur if, for example, two pieces of hardware are attempting to share the same memory address.
- A conflict in DMA addressing has occurred. This can be seen when some display cards conflict with the diskette controller. The symptom is that a `cpio` out to the diskette fails with a parity error panic message.

3. WARNING MESSAGES

The kernel displays a WARNING error message when it encounters a problem that is not serious enough to require stopping the system suddenly. The system administrator should correct the cause of the problem. The following are the most common WARNING messages:

`iget - inode table overflow`

This indicates that the number of entries in the inode table is inadequate. Use `kconfig` to increase the `NINODE` tunable parameter and then build and install a new kernel.

`Swap space running out. Needed` *x* `pages`

This indicates that the maximum total process storage space has been exceeded. You can either:

- Add RAM memory. This will reduce the need for swapping and improve system performance.
- Add swap space to the system. This generally requires reinstalling the system with more swap space allocated. This rather drastic step can be avoided if there is unallocated disk space. Use the command `swap -a` to add another swap device.

`Cannot load floating point emulator`

This indicates that there was trouble loading the floating point emulator. The file `/etc/emulator` may be corrupted or missing.

4. NOTICE MESSAGES

NOTICE messages provide information about the status of the system. In some cases this information can be used avoid problems. Some commonly encountered NOTICE messages include:

`File table overflow`

The number of entries in the file table is inadequate. Use `kconfig` to increase the `NFILE` tunable parameter and build and install a new kernel.

`Unexpected NMI in system mode!`
`Unexpected NMI in user mode!`

This message appears when a Non-Maskable Interrupt (NMI) occurs. This message may be an indication that the hardware is failing.

`getcpages - waiting for` *x* `contiguous pages`

This message means that the kernel is not able to allocate enough contiguous pages of memory and is waiting until it is able to get them. This occurs when the system does not have enough RAM to run the processes requested. This is most common while running INTERACTIVE X11 on a system with only 4 MB of memory, although depending on the number of clients, it may happen on any system configuration.

INDEX

An Introduction to Sendmail

CONTENTS

An Introduction to Sendmail

1. WHAT IS sendmail?

The sendmail program is a general internetwork mail routing facility; it can recognize a variety of network protocols, addressing schemes, and system configurations, as well as perform special delivery options, such as forwarding and aliases. sendmail is the mail routing module of choice for the INTERACTIVE UNIX* Operating System. User correspondence originates with the local mailer program (mail, mailx, Mail, etc.), which then sends it to sendmail for delivery.

1.1 What Does sendmail Do?

Put simply, sendmail receives messages and processes them according to rules that are set up for mail disposition.

Mail can arrive in several ways:

- Using uucp (rmail)
- Using a direct connection (mail, mailx)
- Using the TCP transport protocol (SMTP port)

A single message may pass through many hosts on route to its final destination. As a message is processed, each sendmail module adds several lines to the header; thus the header grows as it passes through each sendmail module. This feature can provide valuable clues when you need to trace the path that a message took through the mail system.

sendmail does one of two things with each message that it receives:

- It sends it to another machine.
- It sends it to another program.

The sendmail program uses a configuration file, called /usr/lib/sendmail.cf, to provide custom information for processing addresses. This file contains sets of site-specific rules for generating both sender and recipient addresses. The

/usr/lib/sendmail.cf configuration file may be frozen to provide faster sendmail startup. Typing the command:

```
$ /usr/lib/sendmail -bz
```

creates /usr/lib/sendmail.fc, also known as the "freezefile" or "frozen configuration file." $HOME/.forward is a local file that sendmail checks for routing instructions.

smail is a public domain program that routes UUCP mail by the lowest-cost path through intervening hosts. This path is determined using a database of host UUCP connections generated by the pathalias program from UUCP maps distributed over the Usenet. If smail is present on a machine, sendmail passes all output destined for UUCP to smail for path routing. If the path database is not built and maintained for the machine, then it is not necessary to install smail.

2. SAMPLE MAIL SYSTEM CONFIGURATIONS

$$\begin{tikzcd} \boxed{\text{mail}} \arrow[r] & \boxed{\text{uucp}} \end{tikzcd}$$

This is the default UNIX System Laboratories (a subsidiary of AT&T) mail configuration.

$$\begin{tikzcd} \boxed{\text{mailx}} \arrow[r] & \boxed{\text{sendmail}} \arrow[r] & \boxed{\text{uucp}} \end{tikzcd}$$

This is a UUCP-only installation where the user is not using `smail`.

$$\begin{tikzcd} \boxed{\text{mailx}} \arrow[r] & \boxed{\text{sendmail}} \arrow[r] & \boxed{\text{tcp}} \end{tikzcd}$$

This is a network-only configuration. Note that TCP should really be thought of as a protocol built into `sendmail` rather than as a separate program.

$$\begin{tikzcd} \boxed{\text{mailx}} \arrow[r] & \boxed{\text{sendmail}} \arrow[r] \arrow[dr] & \boxed{\text{uucp}} \\ & & \boxed{\text{tcp}} \end{tikzcd}$$

This is a network and UUCP configuration where the user is not using `smail`.

$$\begin{tikzcd} \boxed{\text{mailx}} \arrow[r] & \boxed{\text{sendmail}} \arrow[r] \arrow[dr] & \boxed{\text{smail}} \arrow[r] & \boxed{\text{uucp}} \\ & & \boxed{\text{tcp}} & \end{tikzcd}$$

This is a network and UUCP configuration where the user is using `smail`.

3. THE LOCATION OF MAIL-RELATED FILES

Directory	*File Name*	*Comment*
/bin	lmail, rmail, mail, smail	executable binary files
/usr/bin	mailx	executable binary files
/usr/lib	sendmail sendmail.cf, sendmail.fc, sendmail.hf, aliases	the sendmail program various configuration and help files
/usr/lib/mailx	rmmail mailx.help, mailx.help.~	executable binary file mailer help files
/usr/lib/uucp	Permissions, Systems	
/usr/spool/mqueue	syslog	the spool directory for sendmail
/etc/default	smail	the smail configuration file

4. SETTING UP `sendmail`

4.1 Installation of Binaries

Install the Basic Networking package, which contains the correct versions of `mail` and `mailx` to be used with `sendmail` and the `sendmail` binary. Install any networking packages you intend to use, then run the `sysadm setmail` program to configure your mail handling system to suit the specific needs of your machine.

4.2 The `sysadm setmail` Program

Before using the `sysadm setmail` program, you should determine how you want to send mail to other machines. You can choose UUCP (using a modem and phone lines) or a TCP/IP network (using Ethernet* to the destination machine or a gateway machine), or both options can be supported simultaneously on the same machine. Based on your answer, the `setmail` utility copies the appropriate configuration template to the file `/usr/lib/sendmail.cf`. Refer to the "Sendmail Installation Instructions" for information about running the `sysadm setmail` program.

4.3 The Configuration File

`/usr/lib/sendmail.cf` is the configuration file used by `sendmail`. This file is very important! If `sendmail.cf` is not correct, then your mail system will not work. The `sysadm setmail` program modifies this file based on your answers to its questions. You should not need to make further changes to this file unless you have an unusual network configuration. Each configuration option starts with the characters `CF_` so that they are easily located with an editor.

- It is not necessary to set your host name in the configuration file if the command `uname -n` returns the name that you want to use. If you want to set your host name, remove the comment character (#) and change `CF_`*HOST* to the host name you want. You may also want to change the line `Dj$w` to `Dj$w.$D` if you do not put the domain name on the `Dw` line.

- Entries are given for relay hosts to various networks. If you want to exchange mail with sites on these networks, you can put the host name of the appropriate relay site for forwarding mail to each network to which you might send mail.

  There may be provision for a "smart" relay host in the configuration file; this is referred to as `DSCF_SMART`. For example, if you have a smart host named `nell`, substitute `DSnell`. The smart relay host for your mail system should contain the current copies of the `/etc/hosts` and `/usr/lib/aliases` files.

- Check the line starting with `DD` to make sure your domain name is correct. If it is not, rerun the `sysadm setmail` utility rather than editing the file, since the domain information is also stored in other locations.

  For example, if your machine is known as `heep.abc.xyz.com` in the file `/etc/hosts`, use `DDabc.xyz.com`.

- Refreeze the `sendmail` configuration file using the command:

```
$ /usr/lib/sendmail -bz
```

4.4 For Ethernet Usage

The `sendmail` program uses resolver routines to read the `/etc/hosts` file or talk to a name server (if one is active), so that any other host that can be reached directly through TCP/IP or has a name server MX record will resolve to a valid TCP/IP address. If you are using TCP/IP to forward mail to a mail relay host that then sends the mail via UUCP, use UUCP syntax (*host*!*user*) and set up the UUCP relay host name in your `sendmail.cf` file. Otherwise, `sendmail` will probably attempt to connect directly to the destination host and fail. Alternately, `sendmail` rules can be set up to forward to domains that cannot be connected to directly, such as `.COM`, `.MIL`, `.EDU`, `.GOV`, and so on.

4.5 For `uucp` Usage

If you are using `uucp` to talk to remote sites, you also need `/usr/lib/uucp/Systems`, which contains names of recognized uucp sites and the `/usr/lib/uucp/Permissions` file.

4.6 Aliasing

Users on one machine can be given aliases on other machines as if they had local logins, so that you can send mail to them using only their “pseudo” login names rather that having to specify a remote path. Local users who often have mail misaddressed (such as mail addressed to “vicky” rather than “vicki”) can also be given aliases to correct the mail delivery. These aliases are stored in the file `/usr/lib/aliases` as described in *aliases*(5).

To make `sendmail` use new aliases, type the command:

```
    $ /usr/lib/sendmail -bi
```

or use the program `newaliases`, which creates the database form of the alias file for quick access by `sendmail`.

5. CHECKING YOUR MAIL SYSTEM

There are several ways you can check out your new mail system:

- Run `sendmail` in test mode to verify the addresses that will be generated by typing:

  ```
  $ /usr/lib/sendmail -bt
  ```

  or use the command line:

  ```
  $ /usr/lib/sendmail -bt -Cnewfile
  ```

  to try out a new configuration file before compiling it.

  When `sendmail` prompts with:

  ```
  Enter <ruleset> <address>
  ```

  enter `0`, followed by an address to test. Other rulesets can also be tested. For detailed debugging of a ruleset, add the option `-d21.9` to the `sendmail` command line. For even more detail, use `-d21.99`.

  - `local` should appear if the mail destination resides on the same machine.
  - `tcpld` or `tcp` should appear if the addressee lives on another node of Ethernet.

- Print a listing of currently active processes to see what processes are invoked by the local mailer by typing:

  ```
  $ ps -ef
  ```

- Dump the contents of the mail queue to see what comes and goes by typing the command:

  ```
  $ mailq
  ```

- If mail is not being delivered or returned, check `/usr/spool/mqueue` to see if there are any leftover files. Check the end of the `syslog` file for mail error messages. If the file is empty, the `syslog` daemon is probably not running. To start it, `su` to `root` and type:

  ```
  # /etc/syslogd
  ```

- Inspect the returned mail header to find out how far the transmission was able to go. As mentioned earlier, each `sendmail` daemon that processes the mail will add a couple of lines to the header. The added lines will start with `Received:` and include the name of the host. (Note that a header line preceded by a "greater than" symbol (>) was generated by the local mail program, not `sendmail`.)

6. FORWARDING MAIL

Forwarding mail through `sendmail` is similar to forwarding mail at the post office; if you move to another net address, `sendmail` will deliver messages to your new address, even if they are addressed to your old one.

Forwarding should be used in cases where you move to another machine, but continue to have an account on the old machine. Mail that would normally get dropped in your mailbox on the old machine will go to your mailbox on the new machine.

6.1 Examples

Suppose someone sends mail to an old address, `vicky@one`. The local mailer hands the message to `sendmail` running on the machine named `one`. This `sendmail` simply sends the mail using the forwarding information, passing the message on to the next machine, i.e., `mail vicky@two`. There are two ways to accomplish this:

automatic
: Set up a `.forward` file in your home directory.

  For example, if you move to another machine named `artful`, make a `.forward` file on the old machine that contains the new address `vicky@artful`.

  Note that `'Forward to artful!vicky'` cannot be the single first line in the user mailfile `/usr/mail/vicky`; this will not work. This syntax only works with the version of `mail` included in the basic UNIX System Laboratories release.

interactive
: If the sender knows your new address, the letter can be addressed directly to `vicky@artful`.

  No `.forward` file is required for mail sent directly.

7. USER NOTES

7.1 Address Nomenclature

The `!` delimiter forces the use of `uux/uucp`. This style of addressing should therefore only be used for `uucp` addresses. (Note that if you use `!`, you must know the complete path to the destination, including all intervening hosts, unless you use `smail` to route the mail or forward it to a UUCP relay host.)

The domain nomenclature (the @ delimiter, such as `vicky@heep`) should be used instead of `heep!vicky` if the host can be reached by Ethernet.

7.2 Header Generated Using `/bin/mail`

The header generated by `sendmail` when you send mail using `/bin/mail` contains the line `Apparently-To:` instead of `To:`. This can be avoided by sending mail with `/usr/bin/mailx`. The message is delivered successfully in either case.

Sendmail — An Internetwork Mail Router

CONTENTS

Sendmail — An Internetwork Mail Router

*Eric Allman**

Britton-Lee, Inc.
1919 Addison Street, Suite 105
Berkeley, California 94704

ABSTRACT

Routing mail through a heterogenous internet presents many new problems. Among the worst of these is that of address mapping. Historically, this has been handled on an *ad hoc* basis. However, this approach has become unmanageable as internets grow.

`sendmail` acts a unified "post office" to which all mail can be submitted. Address interpretation is controlled by a production system, which can parse both domain-based addressing and old-style *ad hoc* addresses. The production system is powerful enough to rewrite addresses in the message header to conform to the standards of a number of common target networks, including old (NCP/RFC 733) Arpanet, new (TCP/RFC 822) Arpanet, UUCP, and Phonenet. `sendmail` also implements an SMTP server, message queueing, and aliasing.

This article is reprinted (with editorial changes) from the University of California, Berkeley document entitled *SENDMAIL — An Internetwork Mail Router*, with permission of the University of California Regents pursuant to INTERACTIVE's License Agreement for Sendmail Software.

*A considerable part of this work was done while under the employ of the INGRES Project at the University of California at Berkeley.

1. INTRODUCTION

`sendmail` implements a general internetwork mail routing facility, featuring aliasing and forwarding, automatic routing to network gateways, and flexible configuration.

In a simple network, each node has an address, and resources can be identified with a host-resource pair; in particular, the mail system can refer to users using a host-username pair. Host names and numbers have to be administered by a central authority, but usernames can be assigned locally to each host.

In an internet, multiple networks with different characterstics and managements must communicate. In particular, the syntax and semantics of resource identification change. Certain special cases can be handled trivially by *ad hoc* techniques, such as providing network names that appear local to hosts on other networks, as with the Ethernet* at XEROX* PARC. However, the general case is extremely complex. For example, some networks require point-to-point routing, which simplifies the database update problem since only adjacent hosts must be entered into the system tables, while others use end-to-end addressing. Some networks use a left-associative syntax and others use a right-associative syntax, causing ambiguity in mixed addresses.

Internet standards seek to eliminate these problems. Initially, these proposed expanding the address pairs to address triples, consisting of {network, host, resource} triples. Network numbers must be universally agreed upon, and hosts can be assigned locally on each network. The user-level presentation was quickly expanded to address domains, comprised of a local resource identification and a hierarchical domain specification with a common static root. The domain technique separates the issue of physical versus logical addressing. For example, an address given in the form `eric@a.cc.berkeley.edu` describes only the logical organization of the address space.

`sendmail` is intended to help bridge the gap between the totally *ad hoc* world of networks that know nothing of each other and the clean, tightly-coupled world of unique network numbers. It can accept old arbitrary address syntaxes, resolving ambiguities using heuristics specified by the system administrator, as well as domain-based addressing. It helps guide the conversion of message formats between disparate networks. In short, `sendmail` is designed to assist a graceful transition to consistent internetwork addressing schemes.

Section 2 discusses the design goals for `sendmail`. Section 3 gives an overview of the basic functions of the system. In section 4, details of usage are discussed. Section 5 compares `sendmail` to other internet mail routers, and an evaluation of `sendmail` is given in section 6, including future plans.

2. DESIGN GOALS

Design goals for `sendmail` include:

1. Compatibility with the existing mail programs, including Bell version 6 mail, Bell version 7 mail [UNIX83], Berkeley `Mail` [Shoens79], BerkNet mail [Schmidt79], and hopefully UUCP mail [Nowitz78a, Nowitz78b]. ARPANET mail [Crocker77a, Postel77] was also required.
2. Reliability, in the sense of guaranteeing that every message is correctly delivered or at least brought to the attention of a human for correct disposal; no message should ever be completely lost. This goal was considered essential because of the emphasis on mail in our environment. It has turned out to be one of the hardest goals to satisfy, especially in the face of the many anomalous message formats produced by various ARPANET sites. For example, certain sites generate improperly formated addresses, occasionally causing error-message loops. Some hosts use blanks in names, causing problems with UNIX* System mail programs that assume that an address is one word. The semantics of some fields are interpreted slightly differently by different sites. In summary, the obscure features of the ARPANET mail protocol really *are* used and are difficult to support, but must be supported.
3. Existing software to do actual delivery should be used whenever possible. This goal derives as much from political and practical considerations as technical.
4. Easy expansion to fairly complex environments, including multiple connections to a single network type (such as with multiple UUCP or Ethernets [Metcalfe76]). This goal requires consideration of the contents of an address as well as its syntax in order to determine which gateway to use. For example, the ARPANET is bringing up the TCP protocol to replace the old NCP protocol. No host at Berkeley runs both TCP and NCP, so it is necessary to look at the ARPANET host name to determine whether to route mail to an NCP gateway or a TCP gateway.

5. Configuration should not be compiled into the code. A single compiled program should be able to run as is at any site (barring such basic changes as the CPU type or the operating system). We have found this seemingly unimportant goal to be critical in real life. Besides the simple problems that occur when any program gets recompiled in a different environment, many sites like to "fiddle" with anything that they will be recompiling anyway.
6. `sendmail` must be able to let various groups maintain their own mailing lists, and let individuals specify their own forwarding, without modifying the system alias file.
7. Each user should be able to specify which mailer to execute to process mail being delivered for him. This feature allows users who are using specialized mailers that use a different format to build their environment without changing the system and facilitates specialized functions (such as returning an "I am on vacation" message).
8. Network traffic should be minimized by batching addresses to a single host where possible, without assistance from the user.

These goals motivated the architecture illustrated in figure 1.

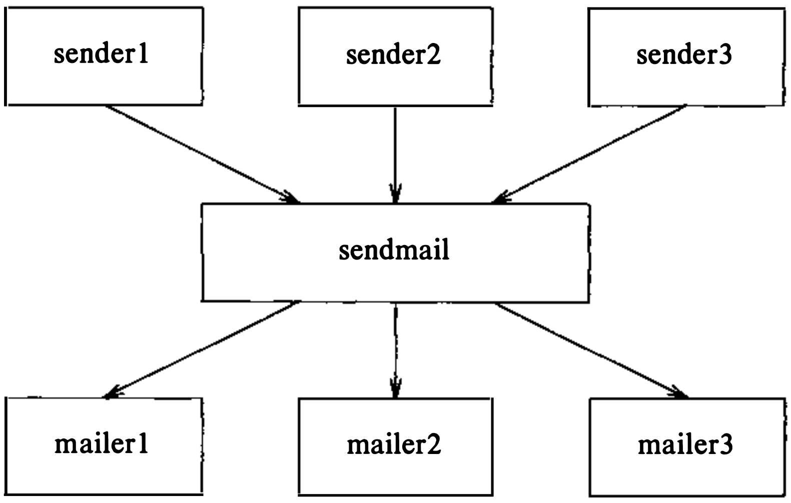


Figure 1. `sendmail` System Structure

The user interacts with a mail generating and sending program. When the mail is created, the generator calls `sendmail`, which routes the message to the correct mailer(s). Since some of the senders may be network servers and some of the mailers may be network clients, `sendmail` may be used as an internet mail gateway.

3. OVERVIEW

3.1 System Organization

sendmail neither interfaces with the user nor does actual mail delivery. Rather, it collects a message generated by a user interface program (UIP) such as Berkeley Mail, MS [Crocker77b], or MH [Borden79], edits the message as required by the destination network, and calls appropriate mailers to do mail delivery or queueing for network transmission.[1] This discipline allows the insertion of new mailers at minimum cost. In this sense sendmail resembles the Message Processing Module (MPM) of [Postel79b].

3.2 Interfaces to the Outside World

There are three ways sendmail can communicate with the outside world, both in receiving and in sending mail. These are using the conventional UNIX System argument vector/return status, speaking SMTP over a pair of UNIX System pipes, and speaking SMTP over an interprocess(or) channel.

3.2.1 Argument Vector/Exit Status

This technique is the standard UNIX System method for communicating with the process. A list of recipients is sent in the argument vector, and the message body is sent on the standard input. Anything that the mailer prints is simply collected and sent back to the sender if there were any problems. The exit status from the mailer is collected after the message is sent, and a diagnostic is printed if appropriate.

3.2.2 SMTP Over Pipes

The SMTP protocol [Postel82] can be used to run an interactive lock-step interface with the mailer. A subprocess is still created, but no recipient addresses are passed to the mailer via the argument list. Instead, they are passed one at a time in commands sent to the processes standard input. Anything appearing on the standard output must be a reply code in a special format.

1. Except when mailing to a file, when sendmail does the delivery directly.

3.2.3 SMTP Over an IPC Connection

This technique is similar to the previous technique, except that it uses a 4.2BSD IPC channel [UNIX83]. This method is exceptionally flexible in that the mailer need not reside on the same machine. It is normally used to connect to a `sendmail` process on another machine.

3.3 Operational Description

When a sender wants to send a message, it issues a request to `sendmail` using one of the three methods described above. `sendmail` operates in two distinct phases. In the first phase, it collects and stores the message. In the second phase, message delivery occurs. If there were errors during processing during the second phase, `sendmail` creates and returns a new message describing the error and/or returns an status code telling what went wrong.

3.3.1 Argument Processing and Address Parsing

If `sendmail` is called using one of the two subprocess techniques, the arguments are first scanned and option specifications are processed. Recipient addresses are then collected, either from the command line or from the SMTP RCPT command, and a list of recipients is created. Aliases are expanded at this step, including mailing lists. As much validation as possible of the addresses is done at this step: syntax is checked, and local addresses are verified, but detailed checking of host names and addresses is deferred until delivery. Forwarding is also performed as the local addresses are verified.

`sendmail` appends each address to the recipient list after parsing. When a name is aliased or forwarded, the old name is retained in the list, and a flag is set that tells the delivery phase to ignore this recipient. This list is kept free from duplicates, preventing alias loops and duplicate messages deliverd to the same recipient, as might occur if a person is in two groups.

3.3.2 Message Collection

`sendmail` then collects the message. The message should have a header at the beginning. No formatting requirements are imposed on the message except that they must be lines of text (i.e., binary data is not allowed). The header is parsed and stored in memory, and the body of the message is saved in a temporary file.

To simplify the program interface, the message is collected even if no addresses were valid. The message will be returned with an error.

3.3.3 Message Delivery

For each unique mailer and host in the recipient list, `sendmail` calls the appropriate mailer. Each mailer invocation sends to all users receiving the message on one host. Mailers that only accept one recipient at a time are handled properly.

The message is sent to the mailer using one of the same three interfaces used to submit a message to `sendmail`. Each copy of the message is prepended by a customized header. The mailer status code is caught and checked, and a suitable error message given as appropriate. The exit code must conform to a system standard or a generic message (`Service unavailable`) is given.

3.3.4 Queueing for Retransmission

If the mailer returned an status that indicated that it might be able to handle the mail later, `sendmail` will queue the mail and try again later.

3.3.5 Return to Sender

If errors occur during processing, `sendmail` returns the message to the sender for retransmission. The letter can be mailed back or written in the file `dead.letter` in the sender's home directory.[2]

3.4 Message Header Editing

Certain editing of the message header occurs automatically. Header lines can be inserted under control of the configuration file. Some lines can be merged; for example, a `From:` line and a `Full-name:` line can be merged under certain circumstances.

2. Obviously, if the site giving the error is not the originating site, the only reasonable option is to mail back to the sender. Also, there are many more error disposition options, but they only affect the error message – the `return to sender` function is always handled in one of these two ways.

3.5 Configuration File

Almost all configuration information is read at runtime from an ASCII file, encoding macro definitions (defining the value of macros used internally), header declarations (telling `sendmail` the format of header lines that it will process specially, i.e., lines that it will add or reformat), mailer definitions (giving information such as the location and characteristics of each mailer), and address rewriting rules (a limited production system to rewrite addresses which is used to parse and rewrite the addresses).

To improve performance when reading the configuration file, a memory image can be provided. This provides a *compiled* form of the configuration file.

4. USAGE AND IMPLEMENTATION

4.1 Arguments

Arguments may be flags and addresses. Flags set various processing options. Following flag arguments, address arguments may be given, unless we are running in SMTP mode. Addresses follow the syntax in RFC822 [Crocker82] for ARPANET address formats. In brief, the format is:

1. Anything in parentheses is thrown away (as a comment).
2. Anything in angle brackets (<>) is preferred over anything else. This rule implements the ARPANET standard that addresses of the form:

   ```
   user name <machine-address>
   ```

   will send to the electronic *machine-address* rather than the human *user name*.
3. Double quotes (") quote phrases; backslashes quote characters. Backslashes are more powerful in that they will cause otherwise equivalent phrases to compare differently – for example, `user` and `"user"` are equivalent, but `\user` is different from either of them.

Parentheses, angle brackets, and double quotes must be properly balanced and nested. The rewriting rules control remaining parsing.[3]

4.2 Mail to Files and Programs

Files and programs are legitimate message recipients. Files provide archival storage of messages, useful for project administration and history. Programs are useful as recipients in a variety of situations, for example, to maintain a public repository of systems messages (such as the news system or the MARS system [Sattley78]).

Any address passing through the initial parsing algorithm as a local address (i.e, not appearing to be a valid address for another mailer) is scanned for two special cases. If prefixed by a vertical bar (|),

3. Disclaimer: Some special processing is done after rewriting local names; see below.

the rest of the address is processed as a shell command. If the user name begins with a slash mark (/), the name is used as a file name instead of a login name.

Files that have `setuid` or `setgid` bits set but no `execute` bits set have those bits honored if `sendmail` is running as `root`.

4.3 Aliasing, Forwarding, Inclusion

`sendmail` reroutes mail three ways. Aliasing applies system wide. Forwarding allows each user to reroute incoming mail destined for that account. Inclusion directs `sendmail` to read a file for a list of addresses, and is normally used in conjunction with aliasing.

4.3.1 Aliasing

Aliasing maps names to address lists using a system-wide file. This file is indexed to speed access. Only names that parse as local are allowed as aliases; this guarantees a unique key (since there are no nicknames for the local host).

4.3.2 Forwarding

After aliasing, recipients that are local and valid are checked for the existence of a `.forward` file in their home directory. If it exists, the message is *not* sent to that user, but rather to the list of users in that file. Often this list will contain only one address, and the feature will be used for network mail forwarding.

Forwarding also permits a user to specify a private incoming mailer. For example, forwarding to:

```
"|/usr/local/newmail myname"
```

will use a different incoming mailer.

4.3.3 Inclusion

Inclusion is specified in RFC733 [Crocker77a] syntax:

```
:Include: pathname
```

An address of this form reads the file specified by `pathname` and sends to all users listed in that file.

The intent is *not* to support direct use of this feature, but rather to use this as a subset of aliasing. For example, an alias of the form:

```
    project: :include:/usr/project/userlist
```

is a method of letting a project maintain a mailing list without interaction with the system administration, even if the alias file is protected.

It is not necessary to rebuild the index on the alias database when a `:include:` list is changed.

4.4 Message Collection

Once all recipient addresses are parsed and verified, the message is collected. The message comes in two parts: a message header and a message body, separated by a blank line.

The header is formatted as a series of lines of the form:

```
    field-name: field-value
```

Field-value can be split across lines by starting the following lines with a space or a tab. Some header fields have special internal meaning, and have appropriate special processing. Other headers are simply passed through. Some header fields may be added automatically, such as time stamps.

The body is a series of text lines. It is completely uninterpreted and untouched, except that lines beginning with a dot have the dot doubled when transmitted over an SMTP channel. This extra dot is stripped by the receiver.

4.5 Message Delivery

The send queue is ordered by receiving host before transmission to implement message batching. Each address is marked as it is sent, so rescanning the list is safe. An argument list is built as the scan proceeds. Mail to files is detected during the scan of the send list. The interface to the mailer is performed using one of the techniques described in section 3.2.

After a connection is established, `sendmail` makes the per-mailer changes to the header and sends the result to the mailer. If any mail is rejected by the mailer, a flag is set to invoke the return-to-sender function after all delivery completes.

4.6 Queued Messages

If the mailer returns a `temporary failure` exit status, the message is queued. A control file is used to describe the recipients to be sent to and various other parameters. This control file is formatted as a series of lines, each describing a sender, a recipient, the time of submission, or some other salient parameter of the message. The header of the message is stored in the control file, so that the associated data file in the queue is just the temporary file that was originally collected.

4.7 Configuration

Configuration is controlled primarily by a configuration file read at startup. `sendmail` should not need to be recomplied except:

1. To change operating systems (V6, V7/32V, 4BSD).
2. To remove or insert the DBM (UNIX System database) library.
3. To change ARPANET reply codes.
4. To add headers fields requiring special processing.

Adding mailers or changing parsing (i.e., rewriting) or routing information does not require recompilation.

If the mail is being sent by a local user, and the file `.mailcf` exists in the sender's home directory, that file is read as a configuration file after the system configuration file. The primary use of this feature is to add header lines.

The configuration file encodes macro definitions, header definitions, mailer definitions, rewriting rules, and options.

4.7.1 Macros

Macros can be used in three ways. Certain macros transmit unstructured textual information into the mail system, such as the name `sendmail` will use to identify itself in error messages. Other macros transmit information from `sendmail` to the configuration file for use in creating other fields (such as argument vectors to mailers), e.g., the name of the sender, and the host and user of the recipient. Other macros are unused internally, and can be used as shorthand in the configuration file.

4.7.2 Header Declarations

Header declarations inform `sendmail` of the format of known header lines. Knowledge of a few header lines is built into `sendmail`, such as the `From:` and `Date:` lines.

Most configured headers will be automatically inserted in the outgoing message if they do not exist in the incoming message. Certain headers are suppressed by some mailers.

4.7.3 Mailer Declarations

Mailer declarations tell `sendmail` of the various mailers available to it. The definition specifies the internal name of the mailer, the path name of the program to call, some flags associated with the mailer, and an argument vector to be used on the call; this vector is macro-expanded before use.

4.7.4 Address Rewriting Rules

The heart of address parsing in `sendmail` is a set of rewriting rules. These are an ordered list of pattern-replacement rules, (somewhat like a production system, except that order is critical), which are applied to each address. The address is rewritten textually until it is either rewritten into a special canonical form (i.e., a (mailer, host, user) 3-tuple, such as {arpanet, usc-isif, postel} representing the address `postel@usc-isif`), or it falls off the end. When a pattern matches, the rule is reapplied until it fails.

The configuration file also supports the editing of addresses into different formats. For example, an address of the form:

```
    ucsfcgl!tef
```

might be mapped into:

```
    tef@ucsfcgl.UUCP
```

to conform to the domain syntax. Translations can also be done in the other direction.

4.7.5 Option Setting

There are several options that can be set from the configuration file. These include the path names of various support files, timeouts, default modes, etc.

5. COMPARISON WITH OTHER MAILERS

5.1 delivermail

sendmail is an outgrowth of delivermail. The primary differences are:

1. Configuration information is not compiled in. This change simplifies many of the problems of moving to other machines. It also allows easy debugging of new mailers.
2. Address parsing is more flexible. For example, delivermail only supported one gateway to any network, whereas sendmail can be sensitive to host names and reroute to different gateways.
3. Forwarding and :include: features eliminate the requirement that the system alias file be writable by any user (or that an update program be written, or that the system administrator make all changes).
4. sendmail supports message batching across networks when a message is being sent to multiple recipients.
5. A mail queue is provided in sendmail. Mail that cannot be delivered immediately but can potentially be delivered later is stored in this queue for a later retry. The queue also provides a buffer against system crashes; after the message has been collected it may be reliably redelivered even if the system crashes during the initial delivery.
6. sendmail uses the networking support provided by 4.2BSD to provide a direct interface networks such as the ARPANET and/or Ethernet using SMTP (the Simple Mail Transfer Protocol) over a TCP/IP connection.

5.2 MMDF

MMDF [Crocker79] spans a wider problem set than sendmail. For example, the domain of MMDF includes a phone network mailer, whereas sendmail calls on preexisting mailers in most cases.

MMDF and sendmail both support aliasing, customized mailers, message batching, automatic forwarding to gateways, queueing, and retransmission. MMDF supports two-stage timeout, which sendmail does not support.

The configuration for MMDF is compiled into the code.[4]

Since MMDF does not consider backwards compatibility as a design goal, the address parsing is simpler but much less flexible.

It is somewhat harder to integrate a new channel[5] into MMDF. In particular, MMDF must know the location and format of host tables for all channels, and the channel must speak a special protocol. This allows MMDF to do additional verification (such as verifying host names) at submission time.

MMDF strictly separates the submission and delivery phases. Although `sendmail` has the concept of each of these stages, they are integrated into one program, whereas in MMDF they are split into two programs.

5.3 Message Processing Module

The Message Processing Module (MPM) discussed by Postel [Postel79b] matches `sendmail` closely in terms of its basic architecture. However, like MMDF, the MPM includes the network interface software as part of its domain.

MPM also postulates a duplex channel to the receiver, as does MMDF, thus allowing simpler handling of errors by the mailer than is possible in `sendmail`. When a message queued by `sendmail` is sent, any errors must be returned to the sender by the mailer itself. Both MPM and MMDF mailers can return an immediate error response, and a single error processor can create an appropriate response.

MPM prefers passing the message as a structured object, with type-length-value tuples.[6] Such a convention requires a much higher degree of cooperation between mailers than is required by `sendmail`. MPM also assumes a universally agreed upon internet name space (with each address in the form of a net-host-user tuple), which `sendmail` does not.

4. Dynamic configuration tables are currently being considered for MMDF, allowing the installer to select either compiled or dynamic tables.

5. The MMDF equivalent of a `sendmail mailer`.

6. This is similar to the NBS standard.

6. EVALUATIONS AND FUTURE PLANS

`sendmail` is designed to work in a nonhomogeneous environment. Every attempt is made to avoid imposing unnecessary constraints on the underlying mailers. This goal has driven much of the design. One of the major problems has been the lack of a uniform address space, as postulated in [Postel79a] and [Postel79b].

A nonuniform address space implies that a path will be specified in all addresses, either explicitly (as part of the address) or implicitly (as with implied forwarding to gateways). This restriction has the unpleasant effect of making replying to messages exceedingly difficult, since there is no one *address* for any person, but only a way to get there from wherever you are.

Interfacing to mail programs that were not initially intended to be applied in an internet environment has been amazingly successful, and has reduced the job to a manageable task.

`sendmail` has knowledge of a few difficult environments built in. It generates ARPANET FTP/SMTP compatible error messages (prepended with three-digit numbers [Neigus73, Postel74, Postel82]) as necessary, optionally generates UNIX System-style `From:` lines on the front of messages for some mailers, and knows how to parse the same lines on input. Also, error handling has an option customized for BerkNet.

The decision to avoid doing any type of delivery where possible (even, or perhaps especially, local delivery) has turned out to be a good idea. Even with local delivery, there are issues of the location of the mailbox, the format of the mailbox, the locking protocol used, etc., that are best decided by other programs. One surprisingly major annoyance in many internet mailers is that the location and format of local mail is built in. The feeling seems to be that local mail is so common that it should be efficient. This feeling is not born out by our experience; on the contrary, the location and format of mailboxes seems to vary widely from system to system.

The ability to automatically generate a response to incoming mail (by forwarding mail to a program) seems useful ("I am on vacation until late August ...") but can create problems such as forwarding loops (two people on vacation whose programs send notes back and forth, for instance) if these programs are not well written. A program could be written to do standard tasks correctly, but this would solve the general case.

It might be desirable to implement some form of load limiting. I am unaware of any mail system that addresses this problem, nor am I aware of any reasonable solution at this time.

The configuration file is currently practically inscrutable; considerable convenience could be realized with a higher-level format.

It seems clear that common protocols will be changing soon to accommodate changing requirements and environments. These changes will include modifications to the message header (e.g., [NBS80]) or to the body of the message itself (such as for multimedia messages [Postel80]). Experience indicates that these changes should be relatively trivial to integrate into the existing system.

In tightly coupled environments, it would be nice to have a name server such as Grapevine [Birrell82] integrated into the mail system. This would allow a site such as *Berkeley* to appear as a single host, rather than as a collection of hosts, and would allow people to move transparently among machines without having to change their addresses. Such a facility would require an automatically updated database and some method of resolving conflicts. Ideally this would be effective even without all hosts being under a single management. However, it is not clear whether this feature should be integrated into the aliasing facility or should be considered a "value added" feature outside `sendmail` itself.

As a more interesting case, the CSNET name server [Solomon81] provides an facility that goes beyond a single tightly-coupled environment. Such a facility would normally exist outside of `sendmail` however.

7. ACKNOWLEDGMENTS

Thanks are due to Kurt Shoens for his continual cheerful assistance and good advice, Bill Joy for pointing me in the correct direction (over and over), and Mark Horton for more advice, prodding, and many of the good ideas. Kurt and Eric Schmidt are to be credited for using `delivermail` as a server for their programs (`Mail` and BerkNet respectively) before any sane person should have, and making the necessary modifications promptly and happily. Eric gave me considerable advice about the perils of network software which saved me an unknown amount of work and grief. Mark did the original implementation of the DBM version of aliasing, installed the VFORK code, wrote the current version of `rmail`, and was the person who really convinced me to put the work into `delivermail` to turn it into `sendmail`. Kurt deserves accolades for using `sendmail` when I was myself afraid to take the risk; how a person can continue to be so enthusiastic in the face of so much bitter reality is beyond me.

Kurt, Mark, Kirk McKusick, Marvin Solomon, and many others have reviewed this paper, giving considerable useful advice.

Special thanks are reserved for Mike Stonebraker at Berkeley and Bob Epstein at Britton-Lee, who both knowingly allowed me to put so much work into this project when there were so many other things I really should have been working on.

8. REFERENCES

[1] [Birrell82]
Birrell, A. D., Levin, R., Needham, R. M., and Schroeder, M. D., "Grapevine: An Exercise in Distributed Computing." In *Comm.A.C.M.* 25, 4, April 82.

[2] [Borden79]
Borden, S., Gaines, R. S., and Shapiro, N. Z., *The MH Message Handling System: Users' Manual*. R-2367-PAF. Rand Corporation. October 1979.

[3] [Crocker77a]
Crocker, D. H., Vittal, J. J., Pogran, K. T., and Henderson, D. A. Jr., *Standard for the Format of ARPA Network Text Messages*. RFC733, NIC 41952. In [Feinler78]. November 1977.

[4] [Crocker77b]
Crocker, D. H., *Framework and Functions of the MS Personal Message System*. R-2134-ARPA, Rand Corporation, Santa Monica, California. 1977.

[5] [Crocker79]
Crocker, D. H., Szurkowski, E. S., and Farber, D. J., *An Internetwork Memo Distribution Facility – MMDF*. 6th Data Communication Symposium, Asilomar. November 1979.

[6] [Crocker82]
Crocker, D. H., *Standard for the Format of Arpa Internet Text Messages*. RFC822. Network Information Center, SRI International, Menlo Park, California. August 1982.

[7] [Metcalfe76]
Metcalfe, R., and Boggs, D., "Ethernet: Distributed Packet Switching for Local Computer Networks," *Communications of the ACM 19*, 7. July 1976.

[8] [Feinler78]
Feinler, E., and Postel, J. (eds.), *ARPANET Protocol Handbook*. NIC 7104, Network Information Center, SRI International, Menlo Park, California. 1978.

[9] [NBS80]
National Bureau of Standards, *Specification of a Draft Message Format Standard.* Report No. ICST/CBOS 80-2. October 1980.

[10] [Neigus73]
Neigus, N., *File Transfer Protocol for the ARPA Network.* RFC542, NIC 17759. In [Feinler78]. August 1973.

[11] [Nowitz78a]
Nowitz, D. A., and Lesk, M. E., "A Dial-Up Network of UNIX Systems," Bell Laboratories. In *UNIX Programmer's Manual*, Seventh Edition, Volume 2. August 1978.

[12] [Nowitz78b]
Nowitz, D. A., "Uucp Implementation Description," Bell Laboratories. In *UNIX Programmer's Manual*, Seventh Edition, Volume 2. October 1978.

[13] [Postel74]
Postel, J., and Neigus, N., Revised FTP Reply Codes. RFC640, NIC 30843. In [Feinler78]. June 1974.

[14] [Postel77]
Postel, J., *Mail Protocol.* NIC 29588. In [Feinler78]. November 1977.

[15] [Postel79a]
Postel, J., *Internet Message Protocol.* RFC753, IEN 85. Network Information Center, SRI International, Menlo Park, California. March 1979.

[16] [Postel79b]
Postel, J. B., *An Internetwork Message Structure.* In *Proceedings of the Sixth Data Communications Symposium*, IEEE. New York. November 1979.

[17] [Postel80]
Postel, J. B., *A Structured Format for Transmission of Multi-Media Documents.* RFC767. Network Information Center, SRI International, Menlo Park, California. August 1980.

[18] [Postel82]
Postel, J. B., *Simple Mail Transfer Protocol.* RFC821 (obsoleting RFC788). Network Information Center, SRI International, Menlo Park, California. August 1982.

[19] [Schmidt79]
Schmidt, E., *An Introduction to the Berkeley Network*. University of California, Berkeley California. 1979.

[20] [Shoens79]
Shoens, K., "Mail Reference Manual," University of California, Berkeley. In *UNIX Programmer's Manual*, Seventh Edition, Volume 2C. December 1979.

[21] [Sluizer81]
Sluizer, S., and Postel, J. B., *Mail Transfer Protocol*. RFC780. Network Information Center, SRI International, Menlo Park, California. May 1981.

[22] [Solomon81]
Solomon, M., Landweber, L., and Neuhengen, D., *The Design of the CSNET Name Server*. CS-DN-2, University of Wisconsin, Madison. November 1981.

[23] [Su82]
Su, Zaw-Sing, and Postel, Jon, *The Domain Naming Convention for Internet User Applications*. RFC819. Network Information Center, SRI International, Menlo Park, California. August 1982.

[24] [UNIX83]
The UNIX Programmer's Manual, Seventh Edition, Virtual VAX-11 Version, Volume 1. Bell Laboratories, modified by the University of California, Berkeley, California. March 1983.

Sendmail Installation Instructions

CONTENTS

Sendmail Installation Instructions

1. GETTING STARTED

This document assumes that you have a basic understanding of mail system components and the skills required of a system administrator.

Before installing `sendmail`, you should already have installed any underlying support and networking packages you require. For example, if you intend to use UUCP and TCP/IP connections, you must have the Basic Networking subset and INTERACTIVE TCP/IP extension running on your system.

You should not connect to a network and start sending mail until you have talked to the administrators of that network and fully understand the responsibilities of network hosts.

Mail messages that are sent using this system will contain the network node name of the machine from which the message is sent. For compatibility, node names should be seven or fewer characters. If you are connecting to a public network of any type, it is very important that your node name not conflict with existing node names at sites on the global wide-area network. Non-unique node names can result in the misrouting of electronic mail. Be sure to check for existing names on the network that might conflict with your system's node name. If a conflict exists, select another node name for your machine.

For more information about establishing a UUCP network, refer to the excellent handbook published by O'Reilly and Associates, Inc., *Managing UUCP and Usenet*. They can be contacted at 1-800-338-6887.

2. INSTALLING AND CONFIGURING THE MAIL SYSTEM USING `sysadm`

For information about installing optional subsets, such as the `sendmail` package, refer to section 6.1 of the "INTERACTIVE UNIX Operating System Installation Instructions" in this guide.

After using the `sysadm` system administration program to load the necessary software packages on your system, use `sysadm` to configure your mail software. `sysadm` allows you to select one of four template files that sets up the default values for the method of mail delivery. This file becomes your `sendmail.cf` configuration file after you tailor the values for your system. Defaults are set only for components of your particular mail system.

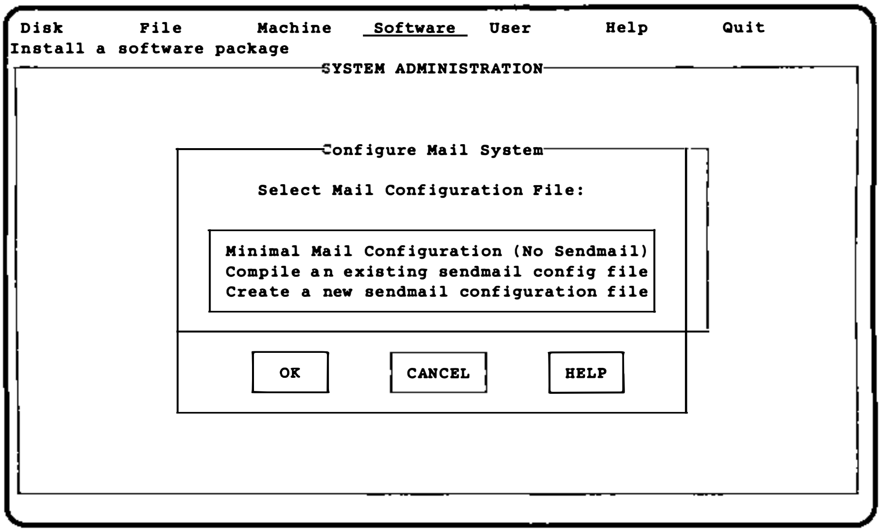
For more information about using `sysadm`, refer to sections 2 and 3 in the "INTERACTIVE UNIX Operating System Maintenance Procedures" in this guide.

Note that the sample installation presented here illustrates only one of the possible choices; the mail delivery method you select may differ from the one shown.

1. Log in as `root` and type `sysadm`. Select the `Mail System Setup` option on the `Software` menu to access the form for configuring your mail system. Or, to bypass the menus, simply type:

   ```
   # sysadm setmail
   ```

   In either case, your screen will look similar to this:

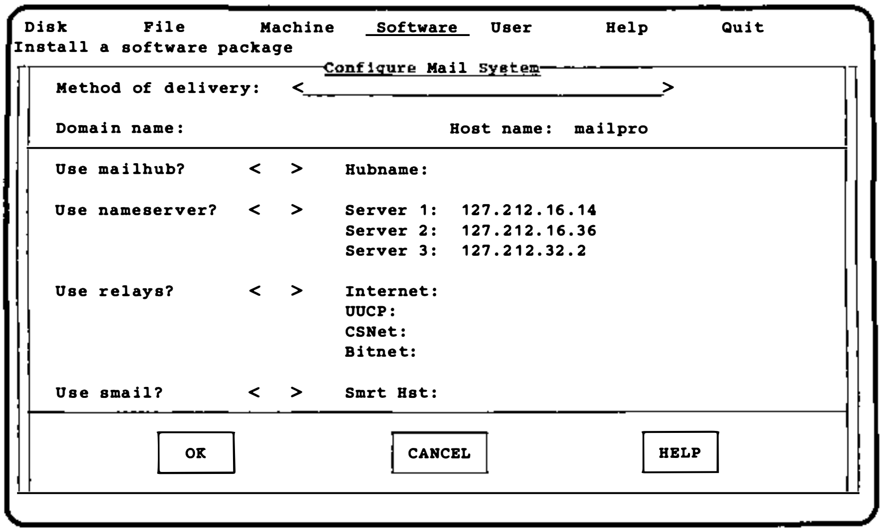


```
 Disk        File        Machine     Software    User        Help        Quit
Install a software package
                              SYSTEM ADMINISTRATION

                    Configure Mail System

                 Select Mail Configuration File:

              Minimal Mail Configuration (No Sendmail)
              Compile an existing sendmail config file
              Create a new sendmail configuration file

                   OK          CANCEL          HELP
```

2. Use the up and down arrow keys to select a minimal mail configuration, to compile the existing `sendmail` configuration file, or to create a new configuration file. TAB to the `OK` button and press ENTER when you have made your selection.

   If you choose to use an existing sendmail configuration file, no further actions are required; the system simply recompiles the existing file.

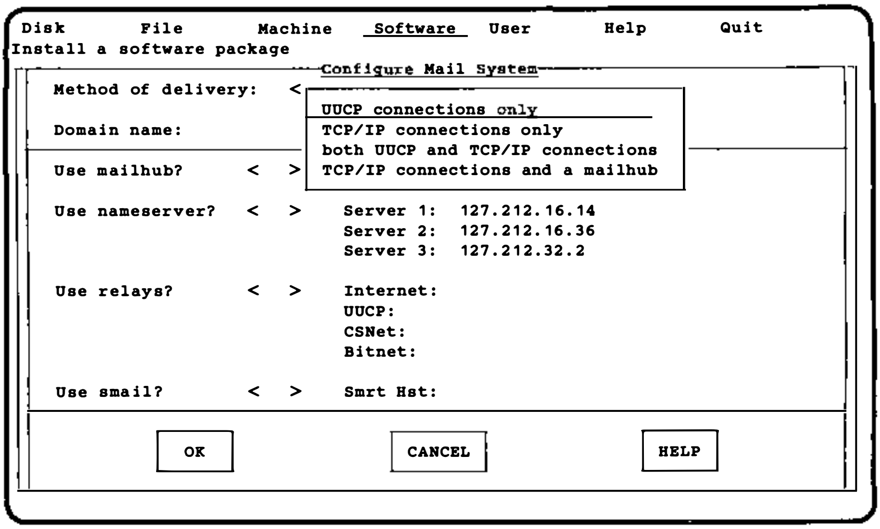
   If you select `Create a new sendmail configuration file`, your screen will look similar to this:



```
Disk        File        Machine     Software    User        Help        Quit
Install a software package
                         Configure Mail System
  Method of delivery:   <                                     >

  Domain name:                         Host name:  mailpro

  Use mailhub?        <   >    Hubname:

  Use nameserver?     <   >    Server 1:  127.212.16.14
                               Server 2:  127.212.16.36
                               Server 3:  127.212.32.2

  Use relays?         <   >    Internet:
                               UUCP:
                               CSNet:
                               Bitnet:

  Use smail?          <   >    Smrt Hst:

              OK                CANCEL               HELP
```

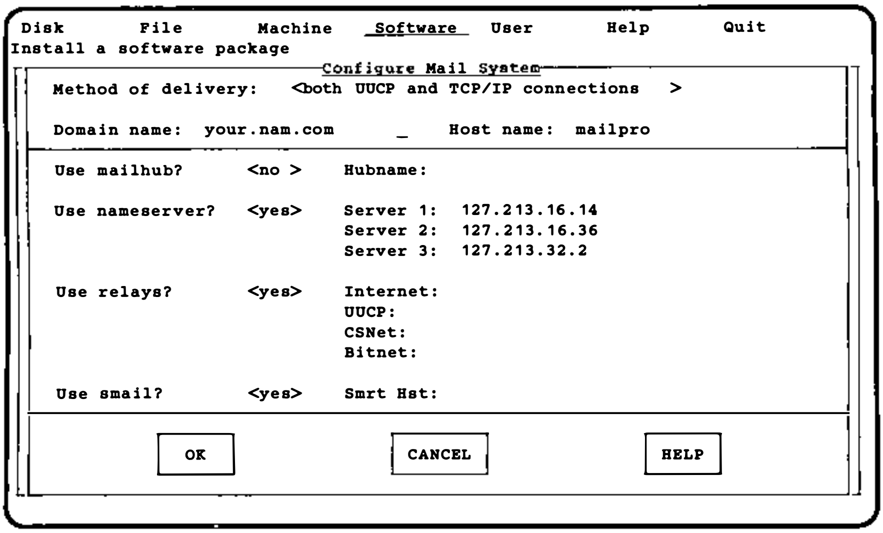
Note that on some systems the system automatically fills in the domain name and host name for your system.

3. Press the spacebar while on the first field to pop up the list of available methods of delivery. Your screen will look similar to this:



```
Disk        File        Machine     Software    User        Help        Quit
Install a software package
                         Configure Mail System
  Method of delivery:   <
                         UUCP connections only
  Domain name:           TCP/IP connections only
                         both UUCP and TCP/IP connections
  Use mailhub?        <   >TCP/IP connections and a mailhub

  Use nameserver?     <   >    Server 1:  127.212.16.14
                               Server 2:  127.212.16.36
                               Server 3:  127.212.32.2

  Use relays?         <   >    Internet:
                               UUCP:
                               CSNet:
                               Bitnet:

  Use smail?          <   >    Smrt Hst:

              OK                CANCEL               HELP
```

4. Use the up and down arrow keys to highlight a method of mail delivery, and press ENTER to select it. For example, if you select `both UUCP and TCP/IP connections`, your screen will look similar to this:



```
Disk        File        Machine     Software    User        Help        Quit
Install a software package
                           Configure Mail System
   Method of delivery:    <both UUCP and TCP/IP connections    >

   Domain name:  your.nam.com       _     Host name:  mailpro

   Use mailhub?       <no >     Hubname:

   Use nameserver?    <yes>     Server 1:   127.213.16.14
                                Server 2:   127.213.16.36
                                Server 3:   127.213.32.2

   Use relays?        <yes>     Internet:
                                UUCP:
                                CSNet:
                                Bitnet:

   Use smail?         <yes>     Smrt Hst:

              OK                CANCEL                HELP
```

If you have a network that is organized by domains, you should already have set your domain. Type in the name of your domain (do not include your host name or the dot), and press ENTER to leave the `Domain name:` field. For example, `isc.com`, is the domain registered to INTERACTIVE Systems Corporation. The `com` suffix is used by commercial companies, as distinguished from educational institutions (`edu`), the government (`gov`), or military (`mil`) sites. As another example, `uucp` is a pseudo-domain that describes the "network" used to reach the site. Using a pseudo-domain is not recommended because many existing mail sites in the UUCP network do not understand this notation and mail sent using this method may fail.

The domain information is stored in two places: the `sendmail` configuration file, `/usr/lib/sendmail.cf`, and the BIND resolver configuration file, `/etc/resolv.conf`. If you want to change your domain name later, you should use `sysadm setmail` to do so. See your network administrator or refer to the "Name Server Operations Guide for

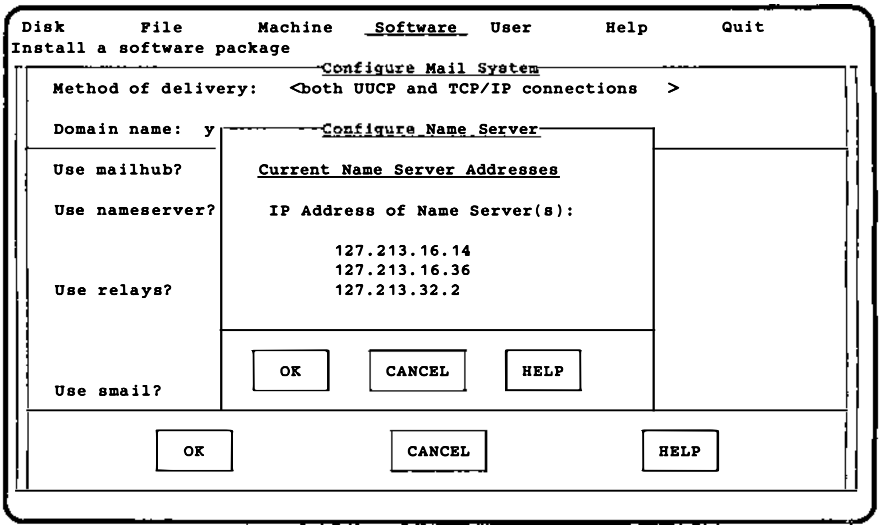
BIND," in your *INTERACTIVE TCP/IP Guide* for a detailed discussion of domains.

☛ Do not connect to a network and start sending mail until you have talked to the administrators of that network and are sure you fully understand the responsibilities of network hosts.

Based on the delivery method you chose, a series of additional forms that allow you to tailor the `sendmail` parameters for your system will be displayed.

If the default that appears in a field is `no`, the type of delivery you selected typically does not use that parameter. For example, if UUCP connections are used, then `Use mailhub?` is set to `no` because mailhubs are not used by UUCP. You may want to change this, however, if, for example, you are using UUCP but you do not want to use `smail`.

5. $\boxed{\text{TAB}}$ to the next field you want to change and press the spacebar to cycle through your choices. Press $\boxed{\text{ENTER}}$ to access the next form when you have made your selection. If you choose to use a name server, your screen will look similar to this:

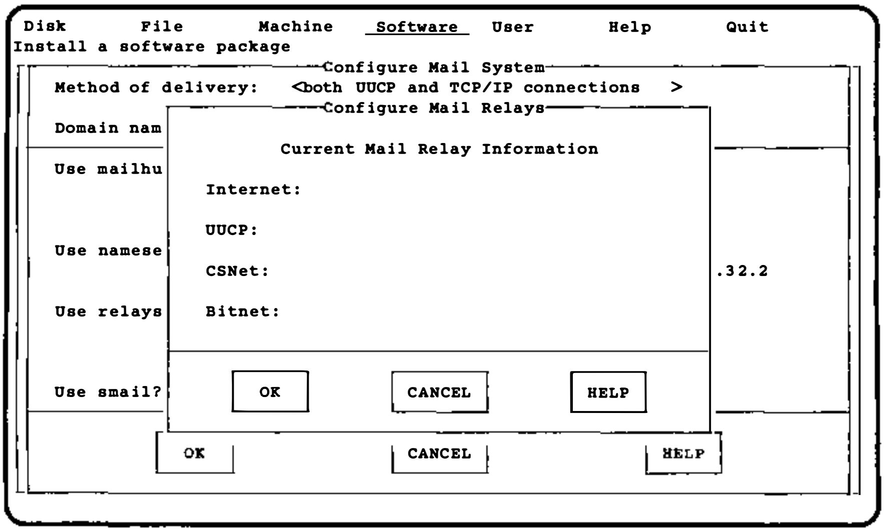


```
Install a software package
                          Configure Mail System
 Method of delivery:   <both UUCP and TCP/IP connections   >

 Domain name:  y        Configure Name Server

 Use mailhub?        Current Name Server Addresses

 Use nameserver?      IP Address of Name Server(s):

                            127.213.16.14
                            127.213.16.36
 Use relays?                127.213.32.2

 Use smail?
```

If you already have a valid `resolv.conf` file, the name server address for your system will already be filled in. If you do not have this file (because, for example, you are building `sendmail` for use on another machine), you should type in the name server address for your configuration.

When you have finished with this form, move to the `OK` button and press ENTER to return to the `Configure Mail System` form.

6. If you choose to use mail relays, your screen will look similar to this:



```
Disk        File        Machine      Software      User        Help        Quit
Install a software package
                          Configure Mail System
  Method of delivery:   <both UUCP and TCP/IP connections    >
                          Configure Mail Relays
  Domain nam
                     Current Mail Relay Information
  Use mailhu
                 Internet:

                 UUCP:
  Use namese
                 CSNet:                                        .32.2

  Use relays     Bitnet:


  Use smail?          OK          CANCEL          HELP

              OK                  CANCEL                  HELP
```

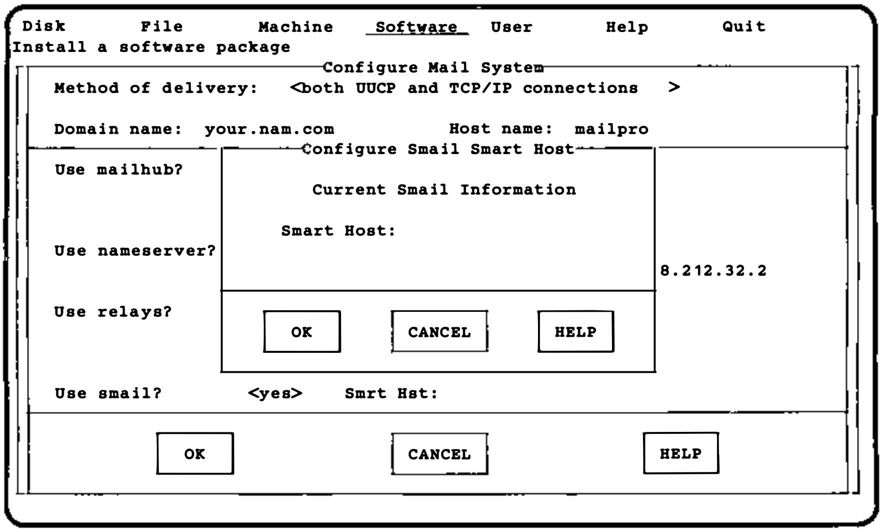
The `sendmail` configuration file installed by this script is taken from a template describing how the system should deliver mail messages. It is not guaranteed to work for your system and may need further tuning. When modifying this file, refer to the "Sendmail Installation and Operation Guide," which describes the format and content of the `sendmail` configuration file. Items that may have to be modified, such as relay site names, are preceded by `CF_` so that they are easy to locate with a text editor.

Relay sites are other hosts to which your machine is connected by a network. These hosts understand how to send mail to other networks, such as BITNET, CSNET, and the Internet.

Any mail messages addressed to domains in other networks that are sent from your system have to be routed by your system or by a relay site.

7. Type in the appropriate information for your system and press $\boxed{\text{ENTER}}$ to move from field to field. When you have finished entering the information for your system, move to the `OK` button and press $\boxed{\text{ENTER}}$ to return to the `Configure Mail System` form.

8. If you choose to use `smail`, your screen will look similar to this:



```
 Disk       File       Machine    Software   User       Help       Quit
Install a software package
                          Configure Mail System
  Method of delivery:   <both UUCP and TCP/IP connections   >

  Domain name:  your.nam.com          Host name:  mailpro
                       Configure Smail Smart Host
  Use mailhub?
                        Current Smail Information

                      Smart Host:
  Use nameserver?
                                                        8.212.32.2

  Use relays?              OK        CANCEL       HELP

  Use smail?       <yes>    Smrt Hst:

              OK                 CANCEL                 HELP
```

If are planning to use a `uucp` connection, refer to *smail*(8) for more information about `smail` and its use.

If you do not use `smail`, `sendmail` will call `uux` directly when sending mail messages via `uucp`. Using `smail` causes `sendmail` to pass `uucp` mail messages to `smail` for automatic `uucp` path routing. The `smail` program generally uses a path database that contains explicit paths to other machines on the UUCP network. This permits `smail` to automatically determine the best route (path) from your machine to another UUCP host. If this database is not available, `smail` can be configured to pass all UUCP mail to another "smart host" that will route the mail. If the routing database is not maintained on your system and there is no smart host entry, then `smail` passes the mail to `uux` as it was received.

The `smail` program has a configuration file in the default directory where options can be set. Refer to *smail*(8) for a description of these options.

9. Type in the name of your smart host and press **ENTER** to return to the `Configure Mail System` form. When you are satisfied with all your choices there, move to the `OK` button and press **ENTER**.

   The `sysadm` software then configures your mail system and will automatically "freeze"(compile) the `sendmail.cf` file so that the `sendmail` program starts up more quickly.

   Note that you may also freeze this file manually. To do this, type:

   ```
   # /usr/lib/sendmail -bz
   ```

   The system should detect changes in the configuration file that require "refreezing," but it is best to freeze again after each configuration change.

   Mail aliases are not kept in the freeze file, so changing user aliases in `/usr/lib/aliases` and updating the alias database with "newaliases" does not require you to refreeze the configuration. Refer to *sendmail*(8) and *aliases*(5) for information about mail user aliases.

10. When the system has finished, your mail system is configured and you are returned to the `Software` menu.

Sendmail Installation and Operation Guide

CONTENTS

Sendmail Installation and Operation Guide

Eric Allman

Britton-Lee, Inc.

ABSTRACT

sendmail implements a general purpose internetwork mail routing facility under the UNIX* Operating System. It is not tied to any one transport protocol – its function may be likened to a crossbar switch, relaying messages from one domain into another. In the process, it can do a limited amount of message header editing to put the message into a format that is appropriate for the receiving domain. All of this is done under the control of a configuration file, sendmail.cf (see section 2.2, /usr/lib/sendmail.cf).

Due to the requirements of flexibility for sendmail, the configuration file can seem somewhat unapproachable. However, there are only a few basic configurations for most sites, for which standard configuration files have been supplied. Most other configurations can be built by adjusting an existing configuration file incrementally.

Although sendmail is intended to run without the need for monitoring, it has a number of features that can be used to monitor or adjust the operation under unusual circumstances. These features are described in this document.

Section 1 provides information on installing and configuring sendmail. Section 2 describes files used

---

This article is derived from the University of California, Berkeley document entitled *SENDMAIL Installation and Operation Guide*, with permission of the University of California Regents pursuant to INTERACTIVE's License Agreement for Sendmail Software.

by `sendmail`. Section 3 explains the day-to-day information you should know to maintain your mail system. If you have a relatively normal site, sections 2 and 3 should contain sufficient information for you to maintain `sendmail`. Section 4 has information regarding the command line arguments. Section 5 describes some parameters that may be safely tweaked. Section 6 contains the nitty-gritty information about the configuration file. This section is for people who must write their own configuration file. The appendixes give brief but detailed explanations of a number of features that are not described in the rest of this document.

The references in this document are found in the companion document "Sendmail – An Internetwork Mail Router," included in this guide. It should be read before this document to gain a basic understanding of how the pieces fit together.

1. INTRODUCTION

This article describes `sendmail` terms, procedures, and maintenance. To install and configure INTERACTIVE `sendmail`, follow the steps in the "Sendmail Installation Instructions."

2. FILES USED BY `sendmail`

2.1 `/usr/lib/sendmail`

This is the binary for `sendmail`. It is located in `/usr/lib`.

2.2 `/usr/lib/sendmail.cf`

This is the configuration file that `sendmail` reads when it starts up. The configuration file describes the mailers it knows about, how to parse addresses, how to rewrite the message header, and the settings of various options.

☛ Use the `sysadm setmail` option to produce a `sendmail.cf` file based on one of several standard configurations. Refer to the "Sendmail Installation Instructions" for additional information.

2.3 `/usr/spool/mqueue`

The `/usr/spool/mqueue` directory is created to hold the mail queue. The mode for the directory is 755.

2.4 `/usr/lib/aliases*`

The system aliases are held in three files. The file `/usr/lib/aliases` is the master copy. A sample is given in `/usr/lib/aliases.base`, which includes some aliases that *must* be defined. You should add any aliases that are pertinent to your system to the file `/usr/lib/aliases`.

In operation, `sendmail` looks at a version of these files maintained by the *dbm*(3) routines. These are stored in `/usr/lib/aliases.dir` and `/usr/lib/aliases.pag`. These can initially be created as empty files, but they will have to be initialized promptly:

```
cp /dev/null /usr/lib/aliases.dir
cp /dev/null /usr/lib/aliases.pag
chmod 644 /usr/lib/aliases.*
newaliases
```

2.5 /usr/lib/sendmail.fc

If you intend to install the frozen version of the configuration file (for quick startup), you can create the file /usr/lib/sendmail.fc and initialize it. This optional optimization step for stable systems may be safely skipped:

```
cp /dev/null /usr/lib/sendmail.fc
/usr/lib/sendmail -bz
```

2.6 /etc/rc3.d/S36sendmail

It is necessary to start up the sendmail daemon when your system reboots. This daemon performs two functions: it listens on the SMTP socket for connections (to receive mail from a remote system), and it processes the queue periodically to insure that mail gets delivered when hosts come up. The file /etc/rc3.d/S36sendmail starts sendmail when the system enters run level 3 (multi-user with networking). The line that actually invokes sendmail has two flags: -bd causes it to listen on the SMTP port, and -q30m causes it to run the queue every half hour.

2.7 /usr/lib/sendmail.hf

This is the help file used by the SMTP HELP command.

2.8 /usr/bin/newaliases

If sendmail is invoked as newaliases, it will simulate the -bi flag (i.e., it will rebuild the alias database, regenerating alias files that have been updated; see section 3.4, “The Alias Database”). This should be a link to /usr/lib/sendmail.

2.9 /usr/bin/mailq

If sendmail is invoked as mailq, it will simulate the -bp flag (i.e., sendmail will print the contents of the mail queue; see section 3.3, “The Mail Queue”). This should be a link to /usr/lib/sendmail.

3. NORMAL OPERATIONS

3.1 Quick Configuration Startup

A fast version of the configuration file may be set up by using the `-bz` flag:

```
    /usr/lib/sendmail -bz
```

This creates the file `/usr/lib/sendmail.fc` ("frozen configuration"). This file is an image of `sendmail`'s data space after reading in the configuration file. If this file exists, it is used instead of `/usr/lib/sendmail.cf`. `sendmail.fc` must be rebuilt manually every time `sendmail.cf` is changed.

The frozen configuration file will be ignored if a `-C` flag is specified or if `sendmail` detects that it is out of date. However, the heuristics are not strong so this should not be trusted.

3.2 The System Log

The system log is supported by the *syslogd*(8) program.

3.2.1 Format

Each line in the system log consists of a timestamp, the name of the machine that generated it (for logging from several machines over the Ethernet*), the word `sendmail:`, and a message.

3.2.2 Levels

If you have *syslogd*(8) installed, you will be able to do logging. There is a large amount of information that can be logged. The log is arranged as a succession of levels. At the lowest level, only extremely strange situations are logged. At the highest level, even the most mundane and uninteresting events are recorded for posterity. As a convention, log levels under ten are considered "useful"; log levels above ten are usually for debugging purposes.

A complete description of the log levels is given in section 5.5.

3.3 The Mail Queue

The mail queue should be processed transparently. However, you may find that manual intervention is sometimes necessary. For example, if a major host is down for a period of time, the queue may become clogged. Although `sendmail` ought to recover gracefully when the host comes up, you may find performance unacceptably bad in the meantime.

3.3.1 Printing the Queue

The contents of the queue can be printed using the `mailq` command (or by specifying the `-bp` flag to `sendmail`):

```
mailq
```

This will produce a listing of the queue IDs, the size of the message, the date the message entered the queue, and the sender and recipients.

3.3.2 Format of Queue Files

All queue files have the form x`f`*AA99999* where *AA99999* is the *ID* for this file and the *x* is a type. The types are:

d	The data file. The message body (excluding the header) is kept in this file.
n	This file is created when an ID is being created. It is a separate file to insure that no mail can ever be destroyed due to a race condition. It should exist for no more than a few milliseconds at any given time.
q	The queue control file. This file contains the information necessary to process the job.
t	A temporary file. This is an image of the `qf` file when it is being rebuilt. It should be renamed to a `qf` file very quickly.
x	A transcript file. This file exists during the life of a session and shows everything that happens during that session.

The `qf` file is structured as a series of lines each beginning with a code letter. The lines are as follows:

D	The name of the data file. There may only be one of these lines.
H	A header definition. There may be any number of these lines. The order is important: they represent the order in the final message. These use the same syntax as header definitions in the configuration file.
R	A recipient address. This will normally be completely aliased, but it is actually realiased when the job is processed. There will be one line for each recipient.

S The sender address. There may only be one of these lines.

E An error address. If any such lines exist, they represent the addresses that should receive error messages.

T The job creation time. This is used to compute when to time out the job.

P The current message priority. This is used to order the queue. Higher numbers mean lower priorities. The priority changes as the message sits in the queue. The initial priority depends on the message class and the size of the message.

M A message. This line is printed by the `mailq` command, and it is generally used to store status information. It can contain any text.

As an example, the following is a queue file sent to `mckusick@calder` and `wnj`:

```
DdfA13557
Seric
T404261372
P132
Rmckusick@calder
Rwnj
H?D?date: 23-Oct-82 15:49:32-PDT (Sat)
H?F?from: eric (Eric Allman)
H?x?full-name: Eric Allman
Hsubject: this is an example message
Hmessage-id: <8209232249.13557@UCBARPA.BERKELEY.ARPA>
Hreceived: by UCBARPA.BERKELEY.ARPA (3.227 [10/22/82])
        id A13557; 23-Oct-82 15:49:32-PDT (Sat)
HTo: mckusick@calder, wnj
```

This shows the name of the data file, the person who sent the message, the submission time (in seconds since January 1, 1970), the message priority, the message class, the recipients, and the headers for the message.

3.3.3 Forcing the Queue

`sendmail` should run the queue automatically at intervals. The algorithm is to read and sort the queue, and then to attempt to process all jobs in order. When it attempts to run the job, `sendmail` first checks to see if the job is locked. If so, it ignores the job.

There is no attempt to insure that only one queue processor exists at any time, since there is no guarantee that a job cannot take forever to process. Due to the locking algorithm, it is impossible for one job to freeze the queue. However, an uncooperative recipient host or a program recipient that never returns can accumulate many processes in your system. Unfortunately, there is no way to resolve this without violating the protocol.

In some cases, you may find that a major host going down for a couple of days may create a prohibitively large queue. This will result in `sendmail` spending an inordinate amount of time sorting the queue. This situation can be fixed by moving the queue to a temporary place and creating a new queue. The old queue can be run later when the offending host returns to service.

To do this, it is acceptable to move the entire queue directory:

```
cd /usr/spool
mv mqueue omqueue; mkdir mqueue; chmod 777 mqueue
```

You should then kill the existing daemon (since it will still be processing in the old queue directory) and create a new daemon.

To run the old mail queue, run the following command:

```
/usr/lib/sendmail -oQ/usr/spool/omqueue -q
```

The `-oQ` flag specifies an alternate queue directory, and the `-q` flag says to just run every job in the queue. If you have a tendency toward voyeurism, you can use the `-v` flag to watch what is going on.

When the queue is finally emptied, you can remove the directory:

```
rmdir /usr/spool/omqueue
```

3.4 The Alias Database

The alias database exists in two forms. One is a text form, maintained in the file `/usr/lib/aliases`. The aliases are of the form:

```
name: name1, name2, ...
```

Only local names may be aliased, e.g.,

```
eric@mit-xx: eric@berkeley.EDU
```

will not have the desired effect. Aliases may be continued by starting any continuation lines with a space or a tab. Blank lines and lines beginning with a sharp sign (#) are comments.

The second form is processed by the *dbm*(3) library. This form is found in the files named `/usr/lib/aliases.dir` and `/usr/lib/aliases.pag`. This is the form that `sendmail` actually uses to resolve aliases. This technique is used to improve performance.

3.4.1 Rebuilding the Alias Database

The DBM version of the database may be rebuilt explicitly by executing the command:

```
newaliases
```

This is equivalent to giving `sendmail` the `-bi` flag:

```
/usr/lib/sendmail -bi
```

If the `D` option is specified in the configuration, `sendmail` will rebuild the alias database automatically. Auto-rebuild can be dangerous on heavily loaded machines with large alias files; if it might take more than 5 minutes to rebuild the database, there is a chance that several processes will start the rebuild process simultaneously.

3.4.2 Potential Problems

There are a number of problems that can occur with the alias database. They all result from a `sendmail` process accessing the DBM version while it is only partially built. This can happen under two circumstances: one process accesses the database while another process is rebuilding it, or the process rebuilding the database dies (due to being killed or a system crash) before completing the rebuild.

`sendmail` has two techniques to try to relieve these problems. First, it ignores interrupts while rebuilding the database; this avoids the problem of someone aborting the process leaving a partially rebuilt database. Second, at the end of the rebuild it adds an alias of the form:

```
@: @
```

(which is not normally legal). Before `sendmail` will access the database, it checks to insure that this entry exists[1]. `sendmail` will wait for this entry to appear, at which point it will force a rebuild itself[2].

3.4.3 List Owners

If an error occurs on sending to a certain address, say *x*, `sendmail` will look for an alias of the form `owner-`*x* to receive the errors. This is typically useful for a mailing list where the submitter of the list has no control over the maintenance of the list itself; in this case the list maintainer would be the owner of the list. For example:

```
unix-wizards: eric@ucbarpa, wnj@monet, nosuchuser,
        sam@matisse
owner-unix-wizards: eric@ucbarpa
```

would cause `eric@ucbarpa` to get the error that will occur when someone sends to `unix-wizards` due to the inclusion of `nosuchuser` on the list.

3.5 Per-User Forwarding (`.forward` Files)

As an alternative to the alias database, any user may put a file with the name `.forward` in his or her home directory. If this file exists, `sendmail` redirects mail for that user to the list of addresses listed in the `.forward` file. For example, if the home directory for user `mckusick` has a `.forward` file with contents:

```
mckusick@ernie
kirk@calder
```

then any mail arriving for `mckusick` will be redirected to the specified accounts.

1. The option is required in the configuration for this action to occur. This should normally be specified.
2. Note: The `D` option must be specified in the configuration file for this operation to occur. If the `D` option is not specified, a warning message is generated and `sendmail` continues.

3.6 Special Header Lines

Several header lines have special interpretations defined by the configuration file. Others have interpretations built into `sendmail` that cannot be changed without changing the source code. These built-ins are described here.

3.6.1 Return-Receipt-To:

If this header is sent, a message will be sent to any specified addresses when the final delivery is complete, that is, when successfully delivered to a mailer with the `l` flag (local delivery) set in the mailer descriptor.

3.6.2 Errors-To:

If errors occur anywhere during processing, this header will cause error messages to go to the listed addresses rather than to the sender. This is intended for mailing lists.

3.6.3 Apparently-To:

If a message comes in with no recipients listed in the message (in a To:, Cc:, or Bcc: line), then `sendmail` will add an `Apparently-To:` header line for any recipients it is aware of. This is not put in as a standard recipient line to warn any recipients that the list is not complete.

At least one recipient line is required under RFC822.

4. ARGUMENTS

The complete list of arguments to `sendmail` is described in detail in Appendix A. Some important arguments are described here.

4.1 Queue Interval

The amount of time between forking a process to run through the queue is defined by the `-q` flag. If you run in mode `f` or `a`, this can be relatively large, since it will only be relevant when a host that was down comes back up. If you run in `q` mode, it should be relatively short, since it defines the maximum amount of time that a message may sit in the queue.

4.2 Daemon Mode

If you allow incoming mail over a TCP/IP connection, you should have a daemon running. This should be set by your `/etc/rc3.d/S36sendmail` file using the `-bd` flag. The `-bd` flag and the `-q` flag may be combined in one call:

```
/usr/lib/sendmail -bd -q30m
```

4.3 Forcing the Queue

In some cases you may find that the queue has gotten clogged for some reason. You can force a queue run using the `-q` flag (with no value). It is entertaining to use the `-v` flag (verbose) when this is done to watch what happens:

```
/usr/lib/sendmail -q -v
```

4.4 Debugging

There are a fairly large number of debug flags built into `sendmail`. Each debug flag has a number and a level, where higher levels mean to print out more information. The convention is that levels greater than nine are *absurd*, i.e., they print out so much information that you would not normally want to see them except for debugging that particular piece of code. Debug flags are set using the `-d` option; the syntax is:

```
debug-flag:     -d debug-list
debug-list:     debug-option [ , debug-option ]
debug-option:   debug-range [ . debug-level ]
debug-range:    integer | integer - integer
debug-level:    integer
```

where spaces are for reading ease only. For example:

```
-d12        Set flag 12 to level 1
-d12.3      Set flag 12 to level 3
-d3-17      Set flags 3 through 17 to level 1
-d3-17.4    Set flags 3 through 17 to level 4
```

For a complete list of the available debug flags, you will have to look at the code (they are too dynamic to keep this documentation up to date).

4.5 Trying a Different Configuration File

An alternative configuration file can be specified using the `-C` flag. For example:

```
/usr/lib/sendmail -Ctest.cf
```

uses the configuration file `test.cf` instead of the default `/usr/lib/sendmail.cf`. If the `-C` flag has no value, it defaults to `sendmail.cf` in the current directory.

4.6 Changing the Values of Options

Options can be overridden using the `-o` flag. For example:

```
/usr/lib/sendmail -oT2m
```

sets the `T` (timeout) option to 2 minutes for this run only.

5. TUNING

There are a number of configuration parameters you may want to change, depending on the requirements of your site. Most of these are set using an option in the configuration file. For example, the line `OT3d` sets option `T` to the value `3d` (3 days).

Most of these options default appropriately for most sites. However, sites having very high mail loads may find they need to tune them as appropriate for their mail load. In particular, sites experiencing a large number of small messages, many of which are delivered to many recipients, may find that they need to adjust the parameters dealing with queue priorities.

5.1 Timeouts

All time intervals are set using a scaled syntax. For example, `10m` represents 10 minutes, whereas `2h30m` represents 2 ½ hours. The full set of scales is:

`s`	seconds
`m`	minutes
`h`	hours
`d`	days
`w`	weeks

5.1.1 Queue Interval

The argument to the `-q` flag specifies how often a subdaemon will run the queue. This is typically set to between 15 minutes and 1 hour.

5.1.2 Read Timeouts

It is possible to time out when reading the standard input or when reading from a remote SMTP server. Technically, this is not acceptable within the published protocols. However, it might be appropriate to set it to something large in certain environments (such as an hour). This will reduce the chance of large numbers of idle daemons piling up on your system. This timeout is set using the `r` option in the configuration file.

5.1.3 Message Timeouts

After sitting in the queue for a few days, a message will time out. This is to insure that at least the sender is aware of the inability to send a message. The timeout is typically set to 3 days. This timeout is set using the `T` option in the configuration file.

The time of submission is set in the queue, rather than the amount of time left until timeout. As a result, you can flush messages that have been hanging for a short period by running the queue with a short message timeout. For example:

```
/usr/lib/sendmail -oT1d -q
```

will run the queue and flush anything that is 1 day old.

5.2 Forking During Queue Runs

By setting the `Y` option, `sendmail` will fork before each individual message while running the queue. This will prevent `sendmail` from consuming large amounts of memory, so it may be useful in memory-poor environments. However, if the `Y` option is not set, `sendmail` will keep track of hosts that are down during a queue run, which can improve performance dramatically.

5.3 Queue Priorities

Every message is assigned a priority when it is first instantiated, consisting of the message size (in bytes) offset by the message class times the "work class factor" and the number of recipients times the "work recipient factor." The priority plus the creation time of the message (in seconds since January 1, 1970) are used to order the queue. Higher numbers for the priority mean that the message will be processed later when running the queue.

The message size is included so that large messages are penalized relative to small messages. The message class allows users to send "high priority" messages by including a `Precedence:` field in their message; the value of this field is looked up in the `P` lines of the configuration file. Since the number of recipients affects the amount of load a message presents to the system, this is also included into the priority.

The recipient and class factors can be set in the configuration file using the `y` and `z` options respectively. They default to 1000 (for the recipient factor) and 1800 (for the class factor). The initial priority is:

```
pri = size - (class * z) + (nrcpt * y)
```

(Remember, higher values for this parameter actually mean that the job will be treated with lower priority.)

The priority of a job can also be adjusted each time it is processed (that is, each time an attempt is made to deliver it) using the "work time factor," set by the `Z` option. This is added to the priority, so it normally decreases the precedence of the job, on the grounds that jobs that have failed many times will tend to fail again in the future.

5.4 Delivery Mode

There are a number of delivery modes that `sendmail` can operate in, set by the `d` configuration option. These modes specify how quickly mail will be delivered. Legal modes are:

`i`	deliver interactively (synchronously)
`b`	deliver in background (asynchronously)
`q`	queue only (do not deliver)

There are tradeoffs. Mode `i` passes the maximum amount of information to the sender, but is hardly ever necessary. Mode `q` puts the minimum load on your machine, but means that delivery may be delayed for up to the queue interval. Mode `b` is probably a good compromise. However, this mode can cause large numbers of processes if you have a mailer that takes a long time to deliver a message.

5.5 Log Level

The level of logging can be set for `sendmail.` The default using a standard configuration table is level 9. The levels are as follows:

0	No logging.
1	Major problems only.
2	Message collections and failed deliveries.
3	Successful deliveries.
4	Messages being deferred (due to a host being down, etc.).
5	Normal message queueups.
6	Unusual but benign incidents, e.g., trying to process a locked queue file.
9	Log internal queue ID to external message ID mappings. This can be useful for tracing a message as it travels between several hosts.

12	Several messages that are basically only of interest when debugging.
16	Verbose information regarding the queue.

5.6 File Modes

There are a number of files that may have a number of modes. The modes depend on what functionality you want and the level of security you require.

5.6.1 To `suid` or Not To `suid`?

`sendmail` is installed with *setuid* to `root`. This is safe because, at the point where it is about to *exec*(2) a mailer, it checks to see if the *userid* is zero; if so, it resets the *userid* and *groupid* to a default (set by the `u` and `g` options). (This can be overridden by setting the `S` flag to the mailer for mailers that are trusted and must be called as `root`.)

5.6.2 Temporary File Modes

The mode of all temporary files that `sendmail` creates is determined by the `F` option. Reasonable values for this option are 0600 and 0644. If the more permissive mode is selected, it will not be necessary to run `sendmail` as `root` at all (even when running the queue).

5.6.3 Should My Alias Database Be Writable?

The alias database, `/usr/lib/aliases*`, is supplied with mode 644. If changed to mode `666`, the alias database will be writable by users. There are some dangers inherent in this approach; any user can add himself to any list, or can "steal" any other user's mail.

The database that `sendmail` actually uses is represented by the two files `aliases.dir` and `aliases.pag` (both in `/usr/lib`). The mode on these files should match the mode on `/usr/lib/aliases`. If `aliases` is writable and the DBM files (`aliases.dir` and `aliases.pag`) are not, users will be unable to reflect their desired changes through to the actual database. However, if `aliases` is read-only and the DBM files are writable, a slightly sophisticated user can arrange to steal mail anyway.

If your DBM files are not writable by the world or you do not have auto-rebuild enabled (with the D option), then you must be careful to reconstruct the alias database each time you change the text version:

```
newaliases
```

If this step is ignored or forgotten, any intended changes will also be ignored or forgotten.

6. THE WHOLE SCOOP ON THE CONFIGURATION FILE

This section describes the configuration file in detail, including hints on how to write one of your own if you have to.

There is one point that should be made clear immediately: the syntax of the configuration file is designed to be reasonably easy to parse, since this is done every time `sendmail` starts up, rather than easy for a human to read or write. On the "future project" list is a configuration-file compiler.

An overview of the configuration file is given first, followed by details of the semantics.

6.1 The Syntax

The configuration file is organized as a series of lines, each of which begins with a single character defining the semantics for the rest of the line. Lines beginning with a space or a tab are continuation lines (although the semantics are not well defined in many places). Blank lines and lines beginning with a sharp symbol (#) are comments.

6.1.1 R and S – Rewriting Rules

The core of address parsing are the rewriting rules. These are an ordered production system. `sendmail` scans through the set of rewriting rules looking for a match on the left-hand side (LHS) of the rule. When a rule matches, the address is replaced by the right-hand side (RHS) of the rule.

There are several sets of rewriting rules. Some of the rewriting sets are used internally and must have specific semantics. Other rewriting sets do not have specifically assigned semantics, and may be referenced by the mailer definitions or by other rewriting sets.

The syntax of each of these two commands is:

S*n*

This sets the current ruleset being collected to *n*. If you begin a ruleset more than once, it deletes the old definition.

R*lhs rhs comments*

The fields must be separated by at least one tab character; there may be embedded spaces in the fields. The *lhs* is a pattern that is applied to the input. If it matches, the input is rewritten to the *rhs*. The *comments* are ignored.

6.1.2 D – Define Macro

Macros are named with a single character. These may be selected from the entire ASCII set, but user-defined macros should be selected from the set of uppercase letters only. Lowercase letters and special symbols are used internally.

The syntax for macro definitions is:

```
Dxval
```

where *x* is the name of the macro and *val* is the value it should have. Macros can be interpolated in most places using the escape sequence **$x**.

6.1.3 C and F – Define Classes

Classes of words may be defined to match on the left-hand side of rewriting rules. For example a class of all local names for this site might be created so that attempts to send to oneself can be eliminated. These can either be defined directly in the configuration file or read in from another file. Classes may be given names from the set of uppercase letters. Lowercase letters and special characters are reserved for system use.

The syntax is:

```
Cc word1 word2 ...
Fc file
```

The first form defines the class *c* to match any of the named words. It is permissible to split them among multiple lines; for example, the two forms:

```
CHmonet ucbmonet
```

and

```
CHmonet
CHucbmonet
```

are equivalent. The second form reads the elements of the class *c* from the named *file*.

6.1.4 M – Define Mailer

Programs and interfaces to mailers are defined in this line. The format is:

```
Mname, {field=value} *
```

where *name* is the name of the mailer (used internally only) and the *field=name* pairs define attributes of the mailer. Fields are:

`Path`	The path name of the mailer
`Flags`	Special flags for this mailer
`Sender`	A rewriting set for sender addresses
`Recipient`	A rewriting set for recipient addresses
`Argv`	An argument vector to pass to this mailer
`Eol`	The end-of-line string for this mailer
`Maxsize`	The maximum message length to this mailer

Only the first character of the field name is checked.

6.1.5 H – Define Header

The format of the header lines that `sendmail` inserts into the message is defined by the `H` line. The syntax of this line is:

`H`[*?mflags?*]*hname:htemplate*

Continuation lines in this spec are reflected directly into the outgoing message. The *htemplate* is macro expanded before insertion into the message. If the *mflags* (surrounded by question marks) are specified, at least one of the specified flags must be stated in the mailer definition for this header to be automatically output. If one of these headers is in the input, it is reflected to the output regardless of these flags.

Some headers have special semantics that will be described below.

6.1.6 O – Set Option

There are a number of "random" options that can be set from a configuration file. Options are represented by single characters. The syntax of this line is:

`O`*ovalue*

This sets option `o` to be *value*. Depending on the option, *value* may be a string, an integer, a boolean (with legal values `t`, `T`, `f`, or `F`; the default is TRUE), or a time interval.

6.1.7 T – Define Trusted Users

Trusted users are those users who are permitted to override the sender address using the `-f` flag. These typically are `root`, `uucp`, and `network`, but on some systems it may be convenient to extend this list to include other users, perhaps to support a separate UUCP login for each host. The syntax of this line is:

`T`*user1 user2 . . .*

There may be more than one of these lines.

6.1.8 P – Precedence Definitions

Values for the `Precedence:` field may be defined using the `P` control line. The syntax of this field is:

```
Pname=num
```

When the *name* is found in a `Precedence:` field, the message class is set to *num*. Higher numbers mean higher precedence. Numbers less than zero have the special property that error messages will not be returned. The default precedence is zero. For example, our list of precedences is:

```
Pfirst-class=0
Pspecial-delivery=100
Pjunk=-100
```

6.2 The Semantics

This section describes the semantics of the configuration file.

6.2.1 Special Macros, Conditionals

Macros are interpolated using the construct `$`*x* where *x* is the name of the macro to be interpolated. In particular, lowercase letters are reserved to have special semantics, used to pass information in or out of `sendmail`, and some special characters are reserved to provide conditionals, etc.

Conditionals can be specified using the syntax:

```
$?x text1 $| text2 $.
```

This interpolates *text1* if the macro `$x` is set, and *text2* otherwise. The "else" (`$|`) clause may be omitted.

The following macros *must* be defined to transmit information into `sendmail`:

e	The SMTP entry message
j	The "official" domain name for this site
l	The format of the UNIX System from line
n	The name of the daemon (for error messages)
o	The set of "operators" in addresses
q	Default format of sender address

The `$e` macro is printed out when SMTP starts up. The first word must be the `$j` macro. The `$j` macro should be in RFC821 format. The `$l` and `$n` macros can be considered constants except under terribly unusual circumstances. The `$o` macro consists of a list of characters which will be considered tokens and which will separate tokens when doing parsing. For example, if @ were in the $o macro, then the input a@b would be scanned as three tokens: a, @, and b. Finally, the $q macro specifies how an address should appear in a message when it is defaulted. For example, on our system these definitions are:

```
De$j Sendmail $v ready at $b
DnMAILER-DAEMON
DlFrom $g  $d
Do.:%@!^=/
Dq$g$?x ($x)$.
Dj$H.$D
```

An acceptable alternative for the $q macro is $?x$x $.<$g>. These correspond to the following two formats:

```
eric@Berkeley (Eric Allman)
Eric Allman <eric@Berkeley>
```

Some macros are defined by sendmail for interpolation into argvs for mailers or for other contexts. These macros are:

a	The origination date in Arpanet format
b	The current date in Arpanet format
c	The hop count
d	The date in UNIX System (ctime) format
f	The sender (from) address
g	The sender address relative to the recipient
h	The recipient host
i	The queue ID
p	sendmail's pid
r	Protocol used
s	Sender's host name
t	A numeric representation of the current time
u	The recipient user
v	The version number of sendmail
w	The host name of this site
x	The full name of the sender
z	The home directory of the recipient

There are three types of dates that can be used. The $a and $b macros are in Arpanet format; $a is the time as extracted from the Date: line of the message (if there was one), and $b is the current date and time (used for postmarks). If no Date: line is found in the incoming message, $a is set to the current time also. The $d macro is equivalent to the $a macro in the UNIX System (ctime) format.

The `$f` macro is the ID of the sender as originally determined; when mailing to a specific host, the `$g` macro is set to the address of the sender *relative to the recipient*. For example, if I send to `bollard@matisse` from the machine `ucbarpa`, the `$f` macro will be `eric` and the `$g` macro will be `eric@ucbarpa`.

The `$x` macro is set to the full name of the sender. This can be determined in several ways. It can be passed as a flag to `sendmail`. The second choice is the value of the `Full-name:` line in the header if it exists, and the third choice is the comment field of a `From:` line. If all of these fail, and if the message is being originated locally, the full name is looked up in the `/etc/passwd` file.

When sending, the `$h`, `$u`, and `$z` macros get set to the host, user, and home directory (if local) of the recipient. The first two are set from the `$@` and `$:` part of the rewriting rules, respectively.

The `$p` and `$t` macros are used to create unique strings (e.g., for the `Message-Id:` field). The `$i` macro is set to the queue ID on this host; if put into the timestamp line, it can be extremely useful for tracking messages. The `$v` macro is set to be the version number of `sendmail`; this is normally put in timestamps and has been proven extremely useful for debugging. The `$w` macro is set to the name of this host if it can be determined. The `$c` field is set to the *hop count*, i.e., the number of times this message has been processed. This can be determined by the `-h` flag on the command line or by counting the timestamps in the message.

The `$r` and `$s` fields are set to the protocol used to communicate with `sendmail` and the sending host name; these are not supported in the current version.

6.2.2 Special Classes

The class `$=w` is set to be the set of all names this host is known by. This can be used to delete local host names.

6.2.3 The Left-Hand Side

The left-hand side of rewriting rules contains a pattern. Normal words are simply matched directly. Metasyntax is introduced using a dollar sign. The metasymbols are:

`$*`	Match zero or more tokens
`$+`	Match one or more tokens
`$-`	Match exactly one token

`$=`*x*	Match any token in class *x*
`$~`*x*	Match any token not in class *x*

If any of these match, they are assigned to the symbol **$***n* for replacement on the right-hand side, where *n* is the index in the LHS. For example, if the LHS:

```
$-:$+
```

is applied to the input:

```
UCBARPA:eric
```

the rule will match, and the values passed to the RHS will be:

```
$1      UCBARPA
$2      eric
```

6.2.4 The Right-Hand Side

When the left-hand side of a rewriting rule matches, the input is deleted and replaced by the right-hand side. Tokens are copied directly from the RHS unless they begin with a dollar sign. Metasymbols are:

$*n*	Substitute indefinite token *n* from LHS
$**[*name***$]	Canonicalize *name*
$>*n*	Call ruleset *n*
$#*mailer*	Resolve to *mailer*
$@*host*	Specify *host*
$:*user*	Specify *user*

The `$n` syntax substitutes the corresponding value from a `$+`, `$-`, `$*`, `$=`, or `$~` match on the LHS. It may be used anywhere.

A host name enclosed between `$[` and `$]` is looked up using the *gethostent*(3) routines and replaced by the canonical name. For example, `$[csam$]` would become `lbl-csam.arpa` and in the same manner `$[[128.32.130.2]$]` would be replaced by `vangogh.berkeley.edu`.

The **$>***n* syntax causes the remainder of the line to be substituted as usual and then passed as the argument to ruleset *n*. The final value of ruleset *n* then becomes the substitution for this rule.

The **$#** syntax should *only* be used in ruleset zero. It causes evaluation of the ruleset to terminate immediately, and signals to `sendmail` that the address has completely resolved. The complete syntax is:

$#mailer$@host$: user

This specifies the {mailer, host, user} 3-tuple necessary to direct the mailer. If the mailer is local the host part may be omitted. The *mailer* and *host* must be a single word, but the *user* may be multi-part.

A RHS may also be preceded by a **$@** or a **$:** to control evaluation. A **$@** prefix causes the ruleset to return with the remainder of the RHS as the value. A **$:** prefix causes the rule to terminate immediately, but the ruleset to continue; this can be used to avoid continued application of a rule. The prefix is stripped before continuing.

The **$@** and **$:** prefixes may precede a **$>** spec. For example:

```
R$+        $:$>7$1
```

matches anything, passes that to ruleset seven, and continues; the **$:** is necessary to avoid an infinite loop.

Substitution occurs in the order described, that is, parameters from the LHS are substituted, host names are canonicalized, "subroutines" are called, and finally **$#**, **$@**, and **$:** are processed.

6.2.5 Semantics of Rewriting Rule Sets

There are five rewriting sets that have specific semantics. These are related as depicted by Figure 1.

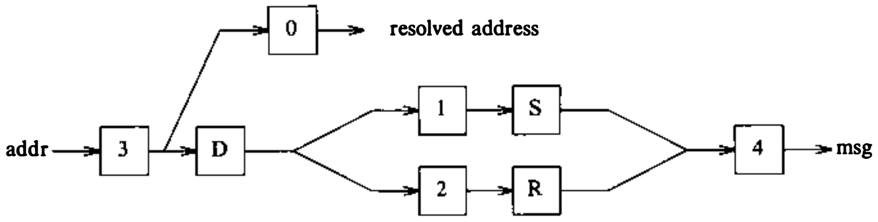


Figure 1. Rewriting Set Semantics

Legend:

D – sender domain addition
S – mailer-specific sender rewriting
R – mailer-specific recipient rewriting

Ruleset three should turn the address into "canonical form." This form should have the basic syntax:

```
local-part@host-domain-spec
```

If no @ sign is specified, then the host-domain-spec *may* be appended from the sender address (if the `C` flag is set in the mailer definition corresponding to the *sending* mailer). Ruleset three is applied by `sendmail` before doing anything with any address.

Ruleset zero is applied after ruleset three to addresses that are going to actually specify recipients. It must resolve to a {*mailer, host, user*} triple. The *mailer* must be defined in the mailer definitions from the configuration file. The *host* is defined into the `$h` macro for use in the `argv` expansion of the specified mailer.

Rulesets one and two are applied to all sender and recipient addresses respectively. They are applied before any specification in the mailer definition. They must never resolve.

Ruleset four is applied to all addresses in the message. It is typically used to translate internal to external form.

6.2.6 Mailer Flags, Etc

There are a number of flags that may be associated with each mailer, each identified by a letter of the alphabet. Many of them are assigned semantics internally. These are detailed in Appendix C. Any other flags may be used freely to conditionally assign headers to messages destined for particular mailers.

6.2.7 The `error` *Mailer*

The mailer with the special name `error` can be used to generate a user error. The (optional) host field is a numeric exit status to be returned, and the user field is a message to be printed. For example, the entry:

```
$#error$:Host unknown in this domain
```

on the RHS of a rule will cause the specified error to be generated if the LHS matches. This mailer is only functional in ruleset zero.

6.3 Building a Configuration File From Scratch

Building a configuration table from scratch is an extremely difficult job. Fortunately, it is almost never necessary to do so; nearly every situation that may come up may be resolved by changing an existing table. In any case, it is critical that you understand what it is that you are trying to do and come up with a philosophy for the configuration table. This section is intended to explain what the real purpose of a configuration table is and to give you some ideas for what your philosophy might be.

6.3.1 What You Are Trying To Do

The configuration table has three major purposes. The first and simplest is to set up the environment for `sendmail`. This involves setting the options, defining a few critical macros, etc. Since these are described in other places, we will not go into more detail here.

The second purpose is to rewrite addresses in the message. This should typically be done in two phases. The first phase maps addresses in any format into a canonical form. This should be done in ruleset three. The second phase maps this canonical form into the syntax appropriate for the receiving mailer. `sendmail` does this in three subphases. Rulesets one and two are applied to all sender and recipient addresses respectively. After this, you may specify per-mailer rulesets for both sender and recipient addresses; this allows mailer-specific customization. Finally, ruleset four is applied to do any default conversion to external form.

The third purpose is to map addresses into the actual set of instructions necessary to get the message delivered. Ruleset zero must resolve to the internal form, which is in turn used as a pointer to a mailer descriptor. The mailer descriptor describes the interface requirements of the mailer.

6.3.2 Philosophy

The particular philosophy you choose will depend heavily on the size and structure of your organization. I will present a few possible philosophies here.

One general point applies to all of these philosophies: it is almost always a mistake to try to do full name resolution. For example, if you are trying to get names of the form `user@host` to the Arpanet, it does not pay to route them to `xyzvax!decvax!ucbvax!c70:user@host` since you then depend on several links not under your control. The best approach to this problem is to simply forward to `xyzvax!user@host` and let `xyzvax` worry about it from there. In summary, just get the message closer to the destination, rather than determining the full path.

6.3.2.1 Large Site, Many Hosts – Minimum Information. Berkeley is an example of a large site, i.e., more than two or three hosts and multiple mail connections. We have decided that the only reasonable philosophy in our environment is to designate one host as the guru for our site. It must be able to resolve any piece of mail it receives. The other sites should have the minimum amount of information they can get away with. In addition, any information they do have should be hints rather than solid information.

For example, a typical site on our local ether network is `monet`. When `monet` receives mail for delivery, it checks whether it knows that the destination host is directly reachable; if so, mail is sent to that host. If it receives mail for any unknown host, it just passes it directly to `ucbvax`, our master host. `ucbvax` may determine that the host name is illegal and reject the message, or may be able to do delivery. However, it is important to note that when a new mail connection is added, the only host that *must* have its tables updated is `ucbvax`; the others *may* be updated if convenient, but this is not critical.

This picture is slightly muddied due to network connections that are not actually located on `ucbvax`. For example, some UUCP connections are currently on `ucbarpa`. However, `monet` *does not* know about this; the information is hidden totally between `ucbvax` and `ucbarpa`. Mail going from `monet` to a UUCP host is transferred via the Ethernet from `monet` to `ucbvax`, then via the Ethernet from `ucbvax` to `ucbarpa`, and then is submitted to UUCP. Although this involves some extra hops, we feel this is an acceptable tradeoff.

An interesting point is that it would be possible to update `monet` to send appropriate UUCP mail directly to `ucbarpa` if the load got too high; if `monet` failed to note a host as connected to `ucbarpa`, it would go via `ucbvax` as before, and if `monet` incorrectly sent a message to `ucbarpa`, it would still be sent by `ucbarpa` to `ucbvax` as before. The only problem that can occur is loops, for example, if `ucbarpa` thought that `ucbvax` had the UUCP connection and vice versa. For this reason, updates should *always* happen to the master host first.

This philosophy results as much from the need to have a single source for the configuration files (typically built using *m4*(1) or some similar tool) as any logical need. Maintaining more than three separate tables by hand is essentially an impossible job.

6.3.2.2 Small Site – Complete Information. A small site (two or three hosts and few external connections) may find it more reasonable to have complete information at each host. This would require that each host know exactly where each network connection is, possibly including the names of each host on that network. As long as the site remains small and the configuration remains relatively static, the update problem will probably not be too great.

6.3.2.3 Single Host. This is in some sense the trivial case. The only major issue is trying to insure that you do not have to know too much about your environment. For example, if you have a UUCP connection, you might find it useful to know about the names of hosts connected directly to you, but this is really not necessary since this may be determined from the syntax.

6.3.3 Relevant Issues

The canonical form you use should almost certainly be as specified in the Arpanet protocols RFC819 and RFC822.

RFC822 describes the format of the mail message itself. `sendmail` follows this RFC closely, to the extent that many of the standards described in this document can not be changed without changing the code. In particular, the following characters have special interpretations:

```
< > ( ) " \
```

Any attempt to use these characters for other than their RFC822 purpose in addresses is probably doomed to disaster.

RFC819 describes the specifics of the domain-based addressing. This is touched on in RFC822 as well. Essentially each host is given a name which is a right-to-left dot qualified pseudo-path from a distinguished root. The elements of the path need not be physical hosts; the domain is logical rather than physical. For example, at Berkeley one legal host might be `a.CC.Berkeley.EDU`; reading from right to left, `EDU` is a top level domain comprising educational institutions, `Berkeley` is a logical domain name, `CC` represents the Computer Center (in this case a strictly logical entity), and `a` is a host in the Computer Center.

Be aware when reading RFC819 that there are a number of errors in it.

6.3.4 How To Proceed

Once you have decided on a philosophy, it is worth examining the available configuration tables to decide if any of them are close enough to steal major parts of. Even under the worst of conditions, there is a fair amount of boiler plate that can be collected safely.

The next step is to build ruleset three. This will be the hardest part of the job. Beware of doing too much to the address in this ruleset, since anything you do will reflect through to the message. In particular, stripping of local domains is best deferred, since this can leave you with addresses with no domain spec at all. Since `sendmail` likes to append the sending domain to addresses with no domain, this can change the semantics of addresses. Also try to avoid fully qualifying domains in this ruleset. Although technically legal, this can lead to unpleasantly and unnecessarily long addresses reflected into messages. The Berkeley configuration files define ruleset nine to qualify domain names and strip local domains. This is called from ruleset zero to get all addresses into a cleaner form.

Once you have ruleset three finished, the other rulesets should be relatively trivial. If you need hints, examine the supplied configuration tables.

6.3.5 Testing the Rewriting Rules — the `-bt` Flag

When you build a configuration table, you can do a certain amount of testing using the `test mode` of `sendmail`. For example, you could invoke `sendmail` as:

```
sendmail -bt -Ctest.cf
```

which would read the configuration file `test.cf` and enter test mode. In this mode, you enter lines of the form:

```
rwset address
```

where `rwset` is the rewriting set you want to use and `address` is an address to apply the set to. Test mode shows you the steps it takes as it proceeds, finally showing you the address it ends up with. You may use a comma separated list of rwsets for sequential application of rules to an input; ruleset three is always applied first. For example:

```
1,21,4 monet:bollard
```

first applies ruleset three to the input "monet:bollard." Ruleset one is then applied to the output of ruleset three, followed similarly by rulesets twenty-one and four.

If you need more detail, you can also use the -d21 flag to turn on more debugging. For example:

```
    sendmail -bt -d21.99
```

turns on an incredible amount of information; a single word address is probably going to print out several pages worth of information.

6.3.6 Building Mailer Descriptions

To add an outgoing mailer to your mail system, you will have to define the characteristics of the mailer.

Each mailer must have an internal name. This can be arbitrary, except that the names `local` and `prog` must be defined.

The path name of the mailer must be given in the `P` field. If this mailer should be accessed via an IPC (TCP/IP) connection, use the string `[IPC]` instead.

The `F` field defines the mailer flags. You should specify an `f` or `r` flag to pass the name of the sender as a `-f` or `-r` flag respectively. These flags are only passed if they were passed to `sendmail`, so that mailers that give errors under some circumstances can be placated. If the mailer is not picky, you can just specify `-f $g` in the `argv` template. If the mailer must be called as `root`, the `S` flag should be given; this will not reset the *userid* before calling the mailer.[3] If this mailer is local (i.e., will perform final delivery rather than another network hop), the `l` flag should be given. Quote characters (backslashes and " marks) can be stripped from addresses if the `s` flag is specified; if this is not given, they are passed through. If the mailer is capable of sending to more than one user on the same host in a single transaction, the `m` flag should be stated. If this flag is on, then the `argv` template containing `$u` will be repeated for each unique user on a given host. The `e` flag will mark the mailer as being `expensive`, which will cause `sendmail` to defer connection until a queue run.[4]

An unusual case is the `C` flag. This flag applies to the mailer that the message is received from, rather than the mailer being sent to; if

3. `sendmail` must be running *setuid* to `root` for this to work.

4. The `c` configuration option must be given for this to be effective.

set, the domain spec of the sender (i.e., the "@host.domain" part) is saved and is appended to any addresses in the message that do not already contain a domain spec. For example, a message of the form:

```
From: eric@ucbarpa
To: wnj@monet, mckusick
```

will be modified to:

```
From: eric@ucbarpa
To: wnj@monet, mckusick@ucbarpa
```

if and only if the `C` flag is defined in the mailer corresponding to `eric@ucbarpa`.

Other flags are described in Appendix C.

The `S` and `R` fields in the mailer description are per-mailer rewriting sets to be applied to sender and recipient addresses respectively. These are applied after the sending domain is appended and the general rewriting sets (numbers one and two) are applied, but before the output rewrite (ruleset four) is applied. A typical use is to append the current domain to addresses that do not already have a domain. For example, a header of the form:

```
From: eric
```

might be changed to be:

```
From: eric@ucbarpa
```

or

```
From: ucbvax!eric
```

depending on the domain it is being shipped into. These sets can also be used to do special purpose output rewriting in cooperation with ruleset four.

The `E` field defines the string to use as an end-of-line indication. A string containing only newline is the default. The usual backslash escapes (\`r`, \n, \`f`, \b) may be used.

Finally, an `argv` template is given as the `E` field. It may have embedded spaces. If there is no `argv` with a `$u` macro in it, `sendmail` will speak SMTP to the mailer. If the path name for this mailer is `[IPC]`, the `argv` should be:

```
IPC $h [ port ]
```

where *port* is the optional port number to connect to.

For example, the specifications:

```
Mlocal, P=/bin/mail, F=rlsm  S=10,   R=20,   A=mail -d $u
Mether, P=[IPC],     F=meC,  S=11,   R=21,   A=IPC $h, M=100000
```

specifies a mailer to do local delivery and a mailer for Ethernet delivery. The first is called `local`, is located in the file `/bin/mail`, takes a picky `-r` flag, does local delivery, quotes should be stripped from addresses, and multiple users can be delivered at once; ruleset ten should be applied to sender addresses in the message and ruleset twenty should be applied to recipient addresses; the `argv` to send to a message will be the word `mail`, the word `-d`, and words containing the name of the receiving user. If a `-r` flag is inserted, it will be between the words `mail` and `-d`. The second mailer is called `ether`, it should be connected to via an IPC connection, it can handle multiple users at once, connections should be deferred, and any domain from the sender address should be appended to any receiver name without a domain; sender addresses should be processed by ruleset eleven and recipient addresses by ruleset twenty-one. There is a 100,000 byte limit on messages passed through this mailer.

Appendix A: COMMAND LINE FLAGS

Arguments must be presented with flags before addresses. The flags are:

`-f`*addr*
: The sender's machine address is *addr*. This flag is ignored unless the real user is listed as a trusted user or if *addr* contains an exclamation point (because of certain restrictions in UUCP).

`-r`*addr*
: An obsolete form of `-f`.

`-h`*cnt* Sets the hop count to *cnt*. This represents the number of times this message has been processed by `sendmail` (to the extent that it is supported by the underlying networks). *cnt* is incremented during processing, and if it reaches MAXHOP (currently 30), `sendmail` throws away the message with an error.

`-F`*name*
: Sets the full name of this user to *name*.

`-n` Do not do aliasing or forwarding.

`-t` Read the header for `To:`, `Cc:`, and `Bcc:` lines, and send to everyone listed in those lists. The `Bcc:` line will be deleted before sending. Any addresses in the argument vector will be deleted from the send list.

`-b`*x* Set operation mode to *x*. Operation modes are:

`m`	Deliver mail (default)
`a`	Run in arpanet mode (see below)
`s`	Speak SMTP on input side
`d`	Run as a daemon
`t`	Run in test mode
`v`	Just verify addresses, don't collect or deliver
`i`	Initialize the alias database
`p`	Print the mail queue
`z`	Freeze the configuration file

The special processing for the ARPANET includes reading the `From:` line from the header to find the sender, printing ARPANET style messages (preceded by three digit reply codes for compatibility with the FTP protocol

[Neigus73, Postel74, Postel77]), and ending lines of error messages with `<CRLF>`.

`-q`*time*
: Try to process the queued up mail. If the time is given, `sendmail` will run through the queue at the specified interval to deliver queued mail; otherwise, it only runs once.

`-C`*file* Use a different configuration file. `sendmail` runs as the invoking user (rather than `root`) when this flag is specified.

`-d`*level*
: Set debugging level.

`-o`*xvalue*
: Set option *x* to the specified *value*. These options are described in Appendix B.

Appendix B: CONFIGURATION OPTIONS

The following options may be set using the `-o` flag on the command line or the `O` line in the configuration file. Many of them cannot be specified unless the invoking user is trusted.

`A`*file*	Use the named *file* as the alias file. If no file is specified, use *aliases* in the current directory.
`a`*N*	If set, wait up to *N* minutes for an `@:@` entry to exist in the alias database before starting up. If it does not appear in *N* minutes, rebuild the database (if the `D` option is also set) or issue a warning.
`B`*c*	Set the blank substitution character to *c*. Unquoted spaces in addresses are replaced by this character.
`c`	If an outgoing mailer is marked as being expensive, do not connect immediately. This requires that queuing be compiled in, since it will depend on a queue run process to actually send the mail.
`d`*x*	Deliver in mode *x*. Legal modes are:

i	Deliver interactively (synchronously)
b	Deliver in background (asynchronously)
q	Just queue the message (deliver during queue run)

`D`	If set, rebuild the alias database if necessary and possible. If this option is not set, `sendmail` will never rebuild the alias database unless explicitly requested using `-bi`.
`e`*x*	Dispose of errors using mode *x*. The values for *x* are:

p	Print error messages (default)
q	No messages, just give exit status
m	Mail back errors
w	Write back errors (mail if user not logged in)
e	Mail back errors and give zero exit stat always

`F`*n*	The temporary file mode, in octal. 644 and 600 are good choices.
`f`	Save UNIX System-style `From` lines at the front of headers. Normally they are assumed redundant and discarded.

`g`*n* Set the default group ID for mailers to run in to *n*.

`H`*file* Specify the help file for SMTP.

`i` Ignore dots in incoming messages.

`L`*n* Set the default log level to *n*.

`M`*xvalue*
: Set the macro *x* to *value*. This is intended only for use from the command line.

`m` Send to me, too, even if I am in an alias expansion.

`N`*netname*
: The name of the home network; ARPA by default. The argument of an SMTP HELO command is checked against *hostname*.`netname`, where *hostname* is requested from the kernel for the current connection. If they do not match, `Received:` lines are augmented by the name that is determined in this manner so that messages can be traced accurately.

`o` Assume that the headers may be in old format, i.e., spaces delimit names. This actually turns on an adaptive algorithm: if any recipient address contains a comma, parenthesis, or angle bracket, it will be assumed that commas already exist. If this flag is not on, only commas delimit names. Headers are always output with commas between the names.

`Q`*dir* Use the named *dir* as the queue directory.

`q`*factor*
: Use *factor* as the multiplier in the map function to decide when to just queue up jobs rather than run them. This value is divided by the difference between the current load average and the load average limit (**x** flag) to determine the maximum message priority that will be sent. Defaults to 10000.

`r`*time* Timeout reads after *time* interval.

`S`*file* Log statistics in the named *file*.

`s`	Be super-safe when running things, i.e., always instantiate the queue file, even if you are going to attempt immediate delivery. `sendmail` always instantiates the queue file before returning control the the client under any circumstances.
`T`*time*	Set the queue timeout to *time*. After this interval, messages that have not been successfully sent will be returned to the sender.
`tS,D`	Set the local time zone name to `S` for standard time and `D` for daylight time; this is only used under version six.
`u`*n*	Set the default *userid* for mailers to *n*. Mailers without the `S` flag in the mailer definition will run as this user.
`v`	Run in verbose mode.
`x`*LA*	When the system load average exceeds *LA*, just queue messages (i.e., don't try to send them).
`X`*LA*	When the system load average exceeds *LA*, refuse incoming SMTP connections.
`y`*fact*	The indicated *fact*or is added to the priority (thus *lowering* the priority of the job) for each recipient, i.e., this value penalizes jobs with large numbers of recipients.
`Y`	If set, deliver each job that is run from the queue in a separate process. Use this option if you are short of memory, since the default tends to consume considerable amounts of memory while the queue is being processed.
`z`*fact*	The indicated *fact*or is multiplied by the message class (determined by the `Precedence:` field in the user header and the `P` lines in the configuration file) and subtracted from the priority. Thus, messages with a higher `Priority:` will be favored.
`Z`*fact*	The *fact*or is added to the priority every time a job is processed. Thus, each time a job is processed, its priority will be decreased by the indicated value. In most environments this should be positive, since hosts that are down are all too often down for a long time.

Appendix C: MAILER FLAGS

The following flags may be set in the mailer description.

`f`	The mailer wants a `-f` *from* flag, but only if this is a network forward operation (i.e., the mailer will give an error if the executing user does not have special permissions).
`r`	Same as `f`, but sends a `-r` flag.
`S`	Do not reset the *userid* before calling the mailer. This would be used in a secure environment where `sendmail` ran as `root`. This could be used to avoid forged addresses. This flag is suppressed if given from an "unsafe" environment (e.g, a user's `mail.cf` file).
`n`	Do not insert a UNIX System-style `From:` line on the front of the message.
`l`	This mailer is local (i.e., final delivery will be performed).
`s`	Strip quote characters off of the address before calling the mailer.
`m`	This mailer can send to multiple users on the same host in one transaction. When a `$u` macro occurs in the `argv` part of the mailer definition, that field will be repeated as necessary for all qualifying users.
`F`	This mailer wants a `From:` header line.
`D`	This mailer wants a `Date:` header line.
`M`	This mailer wants a `Message-Id:` header line.
`x`	This mailer wants a `Full-Name:` header line.
`P`	This mailer wants a `Return-Path:` line.
`u`	Uppercase should be preserved in user names for this mailer.
`h`	Uppercase should be preserved in host names for this mailer.
`A`	This is an Arpanet-compatible mailer, and all appropriate modes should be set.

U This mailer wants UNIX System-style `From` lines with the UUCP System-style `remote from <host>` on the end.

e This mailer is expensive to connect to, so try to avoid connecting normally; any necessary connection will occur during a queue run.

X This mailer wants to use the hidden dot algorithm as specified in RFC821; basically, any line beginning with a dot will have an extra dot prepended (to be stripped at the other end). This insures that lines in the message containing a dot will not terminate the message prematurely.

L Limit the line lengths as specified in RFC821.

P Use the return-path in the SMTP `MAIL FROM:` command rather than just the return address; although this is required in RFC821, many hosts do not process return paths properly.

I This mailer will be speaking SMTP to another `sendmail` – as such it can use special protocol features. This option is not required (i.e., if this option is omitted the transmission will still operate successfully, although perhaps not as efficiently as possible).

C If mail is *received* from a mailer with this flag set, any addresses in the header that do not have an at sign (@) after being rewritten by ruleset three will have the `@domain` clause from the sender tacked on. This allows mail with headers of the form:

```
From: usera@hosta
To: userb@hostb, userc
```

to be rewritten as:

```
From: usera@hosta
To: userb@hostb, userc@hosta
```

automatically.

E Escape lines beginning with `From` in the message with a > sign.

Appendix D: SUMMARY OF SUPPORT FILES

This is a summary of the support files that `sendmail` creates or generates.

`/usr/lib/sendmail`
: The `sendmail` binary.

`/usr/bin/newaliases`
: A link to `/usr/lib/sendmail`; causes the alias database to be rebuilt. Running this program is completely equivalent to giving `sendmail` the `-bi` flag.

`/usr/bin/mailq`
: Prints a listing of the mail queue. This program is equivalent to using the `-bp` flag to `sendmail`.

`/usr/lib/sendmail.cf`
: The configuration file, in textual form.

`/usr/lib/sendmail.fc`
: The configuration file represented as a memory image.

`/usr/lib/sendmail.hf`
: The SMTP help file.

`/usr/lib/aliases`
: The textual version of the alias file.

`/usr/lib/aliases.{pag,dir}`
: The alias file in *dbm*(3) format.

`/usr/spool/mqueue`
: The directory in which the mail queue and temporary files reside.

`/usr/spool/mqueue/qf*`
: Control (queue) files for messages.

`/usr/spool/mqueue/df*`
: Data files.

`/usr/spool/mqueue/tf*`
: Temporary versions of the `qf` files, used during queue file rebuild.

`/usr/spool/mqueue/nf*`
: A file used when creating a unique ID.

`/usr/spool/mqueue/xf*`
: A transcript of the current session.

An Introduction to the C Shell

CONTENTS

An Introduction to the C Shell

William Joy
(revised for 4.3BSD by Mark Seiden)

Computer Science Division
Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, California 94720

ABSTRACT

`csh` is a new command language interpreter for UNIX* Systems. It incorporates good features of other shells and a *history* mechanism similar to the *redo* of INTER-LISP. While incorporating many features of other shells which make writing shell programs (shell scripts) easier, most of the features unique to `csh` are designed more for the interactive UNIX System user.

UNIX System users who have read a general introduction to the system will find a valuable basic explanation of the shell here. Simple terminal interaction with `csh` is possible after reading just section two of this document. The third section describes the shell's capabilities which you can explore after you have begun to become acquainted with the shell. Later sections introduce features which are useful, but not necessary for all users of the shell.

Additional information includes a glossary listing special characters of the shell and terms and commands introduced in this manual.

This article is reprinted from the University of California, Berkeley document entitled *An Introduction to the C shell*, with permission of the University of California Regents pursuant to INTERACTIVE's License Agreement for Berkeley software, Version 6.1. Some modifications and editorial changes have been incorporated.

1. INTRODUCTION

A *shell* is a command language interpreter. `csh` is the name of one particular command interpreter on UNIX Systems. The primary purpose of `csh` is to translate command lines typed at a terminal into system actions, such as invocation of other programs. `csh` is a user program just like any you might write. Hopefully, `csh` will be a very useful program for you in your interactions with the UNIX System.

In addition to this document, you will want to refer to *csh*(1) in the *INTERACTIVE UNIX System User's/System Administrator's Reference Manual*, which provides a full description of all features of the shell and is the definitive reference for questions about the shell.

1.1 Acknowledgments

Numerous people have provided good input about previous versions of `csh` and aided in its debugging and in the debugging of its documentation. I would especially like to thank Michael Ubell who made the crucial observation that history commands could be done well over the word structure of input text, and implemented a prototype history mechanism in an older version of the shell. Eric Allman has also provided a large number of useful comments on the shell, helping to unify those concepts which are present and to identify and eliminate useless and marginally useful features. Mike O'Brien suggested the path name hashing mechanism which speeds command execution. Jim Kulp added the job control and directory stack primitives and added their documentation to this introduction.

2. TERMINAL USAGE OF THE SHELL

2.1 The Basic Notion of Commands

A *shell* in the UNIX System acts mostly as a medium through which other *programs* are invoked. While it has a set of *builtin* functions which it performs directly, most commands cause execution of programs that are, in fact, external to the shell. The shell is thus distinguished from the command interpreters of other systems both by the fact that it is just a user program, and by the fact that it is used almost exclusively as a mechanism for invoking other programs.

Commands in the UNIX System consist of a list of strings or *words* interpreted as a *command name* followed by *arguments.* Thus the command:

```
mail bill
```

consists of two words. The first word `mail` names the command to be executed, in this case the `mail` program which sends messages to other users. The shell uses the name of the command in attempting to execute it for you. It will look in a number of *directories* for a file with the name `mail` which is expected to contain the `mail` program.

The rest of the words of the command are given as *arguments* to the command itself when it is executed. In this case the argument *bill* was also specified, which is interpreted by the `mail` program to be the name of a user to whom mail is to be sent. In normal terminal usage you might use the `mail` command as follows:

```
% mail bill
I have a question about the csh documentation.
My document seems to be missing page 5.
Does a page five exist?
        Bill
EOT
%
```

Here you typed a message to send to *bill* and ended this message with a ^D which sent an end-of-file to the `mail` program. (Here and throughout this document, the notation "^*x*" is to be read "control-*x*" and represents the striking of the *x* key while the $\boxed{\text{CTRL}}$ key is held down.) The `mail` program then echoed the characters EOT and transmitted the message. The percent sign character followed by a space (%) was printed before and after the `mail` command by the shell to indicate that input was needed.

After typing the % prompt, the shell was reading command input from your terminal. You typed a complete command `mail` *bill.* The shell then executed the `mail` program with argument *bill* and went dormant waiting for it to complete. The `mail` program then read input from the terminal until you signaled an end-of-file via typing a ^D after which the shell noticed that `mail` had completed and signaled that it was ready to read from the terminal again by printing another % prompt.

This is the essential pattern of all interaction with the UNIX System through the shell. A complete command is typed at the terminal, the shell executes the command, and when this execution completes, it prompts for a new command. If you run the editor for an hour, the shell will patiently wait for you to finish editing and obediently prompt you again whenever you finish editing.

An example of a useful command you can execute now is the `tset` command, which sets the *erase* and *kill* characters on your terminal – the erase character erases the last character you typed, and the kill character erases the entire line you have entered so far. By default, the erase character is the backspace key (equivalent to ^H), and the kill character is @. The user can always reset the erase character back to the default by typing:

```
    tset –e
```

which tells the program `tset` to set the erase character to `tset`'s default setting for this character (a backspace).

2.2 Flag Arguments

A useful notion in the UNIX System is that of a *flag* argument. While many arguments to commands specify file names or user names, some arguments rather specify an optional capability of the command which you wish to invoke. By convention, such arguments begin with the character hyphen (–). Thus the command:

```
    ls
```

will produce a list of the files in the current *working directory*. The option `-s` is the size option, and:

```
    ls –s
```

causes `ls` to also give, for each file, the size of the file in blocks of 512 characters. The manual entry for each command in the *INTERACTIVE UNIX System User's/System Administrator's Reference Manual* and the *INTERACTIVE SDS Guide and Programmer's Reference Manual* gives the available options for the command. The `ls` command has a large number of useful and interesting options. Most other commands have either no options or only one or two options. It is hard to remember options of commands which are not used very frequently, so most UNIX System utilities perform only one or two functions rather than having a large number of hard-to-remember options.

2.3 Output to Files

Commands that normally read input or write output on the terminal can also be executed with this input and/or output done to a file.

Thus, suppose you wish to save the current date in a file called `now`. The command:

```
date
```

will print the current date on the terminal. This is because the terminal is the default *standard output* for the date command and the date command prints the date on its standard output. The shell lets you *redirect* the *standard output* of a command through a notation using the *metacharacter* redirect (>) and the name of the file where output is to be placed. Thus the command:

```
date > now
```

runs the `date` command such that its standard output is the file `now` rather than the terminal. Thus, this command places the current date and time into the file `now`. It is important to know that the `date` command was unaware that its output was going to a file rather than to the terminal. The shell performed this *redirection* before the command began executing.

One other thing to note here is that the file `now` need not have existed before the `date` command was executed; the shell would have created the file if it did not exist. And if the file did exist? If it had existed previously, these previous contents would have been discarded! A shell variable *noclobber* exists to prevent this from happening accidentally; it is discussed in section 3.2.

The system normally keeps files which you create with > and all other files. Thus the default is for files to be permanent. If you wish to create a file which will be removed automatically, you can begin its name with a pound sign (#) character; this 'scratch' character denotes the fact that the file will be a scratch file.[1] The system will remove such files after a couple of days, or sooner if file space becomes very tight. Thus, in running the `date` command above, you do not really want to save the output forever, so you would more likely do:

```
date > #now
```

2.4 Metacharacters in the Shell

The shell has a large number of special characters (such as >) which indicate special functions. These notations have *syntactic* and *semantic* meaning to the shell. In general, most characters which are neither letters nor digits have special meaning to the shell. You will shortly learn a means of *quotation* which allows you to use *metacharacters* without the shell treating them in any special way.

Metacharacters normally have effect only when the shell is reading input. You need not worry about placing shell metacharacters in a letter you are sending via `mail`, or when you are typing in text or data to some other program. Note that the shell is only reading input when it has prompted with '% ' (although you can type input even before it prompts).

2.5 Input From Files; Pipelines

You learned above how to *redirect* the *standard output* of a command to a file. It is also possible to redirect the *standard input* of a command from a file. This is not often necessary since most commands will read from a file whose name is given as an argument. You can give the command:

1. Note that if your erase character is a #, you will have to precede the # with a backslash (\). The fact that the # character is the old (pre-CRT) standard erase character means that it seldom appears in a file name, and allows this convention to be used for scratch files. If you are using a CRT, your erase character should be a ^H. Setting ^H to be your erase character is discussed in section 2.1.

```
sort < data
```

to run the `sort` command with standard input, where the command normally reads its input, from the file `data`. You would more likely say:

```
sort data
```

letting the `sort` command open the file `data` for input itself since this is less to type.

You should note that if you just typed:

```
sort
```

then the sort program would sort lines from its *standard input*. Since you did not *redirect* the standard input, it would sort lines as you typed them on the terminal until you typed a ^D to indicate an end-of-file.

A most useful capability is the ability to combine the standard output of one command with the standard input of another, i.e., to run the commands in a sequence known as a *pipeline*. For instance, the command:

```
ls −s
```

normally produces a list of the files in a directory with the size of each in blocks of 512 characters. If you are interested in learning which of your files is the largest, you may wish to have this sorted by size rather than by name, which is the default way in which `ls` sorts. You could look at the many options of `ls` to see if there was an option to do this, but would eventually discover that there is not. Instead you can use a couple of simple options of the `sort` command, combining it with `ls` to get what you want.

The `-n` option of sort specifies a numeric sort rather than an alphabetic sort. Thus:

```
ls −s | sort −n
```

specifies that the output of the `ls` command run with the option `-s` is to be *piped* to the command `sort` run with the numeric sort option. This would give you a sorted list of our files by size, but with the smallest first. You could then use the `-r` reverse sort option and the *sed* command in combination with the previous command doing:

```
ls –s | sort –n –r | sed 5q
```

Here you have taken a list of your files sorted alphabetically, each with the size in blocks. You have run this to the standard input of the `sort` command asking it to sort numerically in reverse order (largest first). This output has then been run into the command *sed* which copies its input to the standard output. In this case, you have asked *sed* for the first five lines. Thus, this command gives you the names and sizes of your five largest files.

The notation introduced above is called the *pipe* mechanism. Commands separated by pipe (|) characters are connected together by the shell, and the standard output of each is run into the standard input of the next. The leftmost command in a pipeline will normally take its standard input from the terminal, and the rightmost will place its standard output on the terminal. Other examples of pipelines will be given later when the history mechanism is discussed; one important use of pipes which is illustrated there is the routing of information to the line printer.

2.6 File Names

Many commands to be executed will need the names of files as arguments. UNIX System *path names* consist of a number of *components* separated by slash (/). Each component except the last names a directory in which the next component resides, in effect specifying the *path* of directories to follow to reach the file. Thus the path name:

```
/etc/motd
```

specifies a file in the directory `etc` which is a subdirectory of the `root` directory `/`. Within this directory is the file named `motd`, which stands for 'message of the day.' A *path name* that begins with a slash is said to be an *absolute* path name since it is specified from the absolute top of the entire directory hierarchy of the system (the `root`). *Path names* which do not begin with slash (/) are interpreted as starting in the current *working directory*, which is, by default, your `home` directory and can be changed dynamically by the `cd` change directory command. Such path names are said to be *relative* to the working directory since they are found by starting in the working directory and descending to lower levels of directories for each *component* of the path name. If the path name contains no slashes at all, then the file is contained in the working directory itself and the path name is merely the name of the file in this directory. Absolute path names have no relation to the working directory.

Most file names consist of a number of alphanumeric characters and periods (.). In fact, all printing characters except slash (/) may appear in file names. It is inconvenient to have most non-alphabetic characters in file names because many of these have special meaning to the shell. The character period (.) is not a shell-metacharacter and is often used to separate the *extension* of a file name from the base of the name. Thus:

```
prog.c prog.o prog.errs prog.output
```

are four related files. They share a *base* portion of a name (a base portion being that part of the name that is left when a trailing . and following characters which are not . are stripped off). The file `prog.c` might be the source for a C program, the file `prog.o` the corresponding object file, the file `prog.errs` the errors resulting from a compilation of the program, and the file `prog.output` the output of a run of the program.

If you wished to refer to all four of these files in a command, you could use the notation:

```
prog.*
```

This expression is expanded by the shell, before the command to which it is an argument is executed, into a list of names which begin with `prog.`. The character asterisk (*) here matches any sequence (including the empty sequence) of characters in a file name. The names that match are alphabetically sorted and placed in the *argument list* of the command. Thus the command:

```
echo prog.*
```

will echo the names:

```
prog.c prog.errs prog.o prog.output
```

Note that the names are in sorted order here, and in a different order than you listed them above. The `echo` command receives four words as arguments, even though you only typed one word as an argument directly. The four words were generated by *file name expansion* of the one input word.

Other notations for *file name expansion* are also available. The character question mark (?) matches any single character in a file name. Thus:

```
echo ? ?? ???
```

will echo a line of file names; first those with one-character names, then those with two-character names, and finally those with three-character names. The names of each length will be independently sorted.

Another mechanism consists of a sequence of characters between brackets ([and]). This metasequence matches any single character from the enclosed set. Thus:

```
prog.[co]
```

will match:

```
prog.c prog.o
```

in the example above. You can also place two characters around a hyphen (−) in this notation to denote a range. Thus:

```
chap.[1−5]
```

might match files:

```
chap.1 chap.2 chap.3 chap.4 chap.5
```

if they existed. This is shorthand for:

```
chap.[12345]
```

and otherwise equivalent.

An important point to note is that if a list of argument words to a command (an *argument list*) contains file name expansion syntax, and if this file name expansion syntax fails to match any existing file names, then the shell considers this to be an error and prints a diagnostic:

```
No match.
```

and does not execute the command.

Another very important point is that files with the character period (.) at the beginning are treated specially. Neither * nor ? nor the [ ] mechanism will match it. This prevents accidental matching of the file names dot (.) and dot-dot (..) in the working directory which have special meaning to the system, as well as other files such as `.cshrc` which are not normally visible. The special role of the file `.cshrc` will be discussed later.

Another file name expansion mechanism gives access to the path name of the *home* directory of other users. This notation consists of the character tilde (~) followed by another user's login name. For instance, the word '~bill' would map to the path name `/usr/bill` if the home directory for Bill was `/usr/bill`. Since, on large systems, users may have login directories scattered over many different disk volumes with different prefix directory names, this notation provides a convenient way of accessing the files of other users.

A special case of this notation consists of a tilde (~) alone, e.g. `~/mbox`. This notation is expanded by the shell into the file `mbox` in your *home* directory, i.e., into `/usr/bill/mbox`. This can be very useful if you have used `cd` to change to another directory and have found a file you wish to copy using `cp`. If you give the command:

```
cp thatfile ~
```

and your home directory is `/usr/bill`, the shell will expand the command to:

```
cp thatfile /usr/bill
```

since the home directory is `/usr/bill`.

There is also a mechanism that uses the curly brace characters { and } to generate a set of file names that contain common parts. Unlike mechanisms described previously, the resulting file names are not required to exist. This mechanism, which is used less frequently, is described in section 5.2.

2.7 Quotation

You have already seen a number of metacharacters used by the shell. These metacharacters pose a problem in that you cannot use them directly as parts of words. Thus the command:

```
echo *
```

will not echo the character *. It will either echo a sorted list of file names in the current *working directory* or print the message 'No match' if there are no files in the working directory.

The recommended mechanism for placing characters which are neither numbers, digits, slashes (/), periods (.), nor hyphens (−) in an argument word to a command is to enclose it with single quotation characters (′), that is:

```
echo ′*′
```

One special character, the exclamation point (!), is used by the *history* mechanism of the shell and cannot be *escaped* by placing it within ′ characters. It and the character ′ itself can be preceded by a single backslash (\) to prevent their special meaning. Thus:

```
echo \′\!
```

prints:

```
′!
```

These two mechanisms suffice to place any printing character into a word which is an argument to a shell command. They can be combined, as in:

```
echo \′′*′
```

which prints:

```
′*
```

since the first \ escaped the first ′ and the * was enclosed between ′ characters.

2.8 Terminating Commands

When you are executing a command and the shell is waiting for it to complete, there are several ways to force it to stop. For instance, if you type the command:

```
cat /etc/passwd
```

the system will print a copy of a list of all users of the system on your terminal. This is likely to continue for several minutes unless you stop it. You can send an INTERRUPT (INTR) *signal* to the `cat` command by typing ^C on your terminal.[2] Since `cat` does not

2. On some older UNIX systems the DEL or RUBOUT key has the same effect. "stty −a" will tell you the INTR key value.

take any precautions to avoid or otherwise handle this signal, the INTERRUPT will cause it to terminate. The shell notices that `cat` has terminated and prompts you again with %. If you hit INTERRUPT again, the shell will just repeat its prompt since it handles INTERRUPT signals and chooses to continue to execute commands rather than terminating like `cat` did, which would have the effect of logging you out.

Another way in which many programs terminate is when they get an end-of-file from their standard input. Thus, the `mail` program in the first example above was terminated when you typed a ^D, which generates an end-of-file from the standard input. The shell also terminates when it gets an end-of-file printing 'logout'; the UNIX System then logs you off the system. Since this means that typing ^D too many times can accidentally log off users, the shell has a mechanism for preventing this. This *ignoreeof* variable will be discussed in section 3.2.

If a command has its standard input redirected from a file, then it will normally terminate when it reaches the end of this file. Thus if you execute:

```
mail bill < prepared.text
```

you will not have to type a ^D for the `mail` command to terminate. This is because it read to the end-of-file of `prepared.text` in which you placed a message for 'bill' with an editor program. You could also have done:

```
cat prepared.text | mail bill
```

since the `cat` command would then have written the text through the pipe to the standard input of the `mail` command. When the `cat` command completed, it would have terminated, closing down the pipeline, and the `mail` command would have received an end-of-file from it and terminated. Using a pipe here is more complicated than redirecting input so you would more likely use the first form. These commands could also have been stopped by sending an

INTERRUPT. Another possibility for stopping a command is to suspend its execution temporarily, with the possibility of continuing execution later. This is done by sending a STOP signal (typing ^Z[3]). This signal causes all commands running on the terminal (usually one, but more if a pipeline is executing) to become suspended. The shell notices that the command(s) have been suspended, types 'Stopped,' and then prompts for a new command. The previously executing command has been suspended, but otherwise unaffected by the STOP signal. Any other commands can be executed while the original command remains suspended. The suspended command can be continued using the *fg* command with no arguments. The shell will then retype the command to remind you which command is being continued, and cause the command to resume execution. Unless any input files in use by the suspended command have been changed in the meantime, the suspension has no effect whatsoever on the execution of the command. This feature can be very useful during editing, when you need to look at another file before continuing. An example of command suspension follows:

3. In order to enable job control, you must first issue the command:

```
stty swtch ^Z
```

A standard way of automating this is to place the command in your `.login` file.

```
% mail harold
Someone just copied a big file into my directory and its name is
^Z
Stopped
% ls
funnyfile
prog.c
prog.o
% jobs
[1] + Stopped  mail harold
% fg
mail harold
funnyfile. Do you know who did it?
EOT
%
```

In this example someone was sending a message to Harold and forgot the name of the file he wanted to mention. The `mail` command was suspended by typing ^Z. When the shell noticed that the `mail` program was suspended, it typed 'Stopped' and prompted for a new command. Then the `ls` command was typed to find out the name of the file. The *jobs* command was run to find out which command was suspended. At this time, the *fg* command was typed to continue execution of the `mail` program. Input to the `mail` program was then continued and ended with a ^D, which indicated the end of the message, at which time the `mail` program typed EOT. The *jobs* command will show which commands are suspended. The ^Z should only be typed at the beginning of a line since everything typed on the current line is discarded when a signal is sent from the keyboard. This also happens on INTERRUPT and QUIT signals. More information on suspending jobs and controlling them is given in section 3.6.

If you write or run programs that are not fully debugged, then it may be necessary to terminate them somewhat ungracefully. This can be done by sending them a QUIT signal, sent by typing a ^\. This will usually provoke the shell to produce a message such as:

```
Quit (Core dumped)
```

indicating that a file `core` has been created containing information about the running program's state when it terminated due to the QUIT signal. You can examine this file yourself, or forward information to the maintainer of the program telling him/her where the `core` file is.

If you run background commands (as explained in section 3.6), then these commands will ignore INTERRUPT and QUIT signals at the terminal. To terminate them you must use the `kill` command. See section 3.6 for an example.

If you want to examine the output of a command without having it scroll off the screen as the output of the:

```
cat /etc/passwd
```

command will, you can use the command:

```
more /etc/passwd
```

The `more` program pauses after each complete screenful and displays '--More--'; at this point you can press the space bar to get another screenful, press **RETURN** to get another line, type a question mark (?) to get some help on other commands, or type the letter q to end the `more` program. You can also use `more` as a filter, that is:

```
cat /etc/passwd | more
```

works just like the more simple `more` command above.

For pausing the output of commands not involving `more`, you can use the ^S key to stop the typeout. The typeout will resume when you hit ^Q or any other key, but ^Q is normally used because it only restarts the output and does not become input to the program which is running. This works well on low-speed terminals, but since at 9600 baud it is hard to type ^S and ^Q fast enough to paginate the output nicely, a program like `more` is usually used.

2.9 What Now?

We have so far seen a number of mechanisms of the shell and learned a lot about the way in which it operates. The remaining sections will go yet further into the internals of the shell, but you will surely want to try using the shell before you go any further. To use the C Shell as your login shell, the program `/bin/csh` must be named in the last field of the `/etc/passwd` file entry for your user name. You need to log in to your UNIX System as `root` (the superuser) or `sysadm` (the system administrator) in order to modify `/etc/passwd`. See the "INTERACTIVE UNIX Operating System Maintenance Procedures" for more information on using `sysadm` to modify a user's login. On single-user systems or in single-user mode, it is probably safe to edit the `/etc/passwd` file, changing the last field (after the last colon) to `/bin/csh`. You only have to do this once; it takes effect the *next* time you log in. You are now ready to try using `csh`.

Until you change it, the shell you are using when you log into the system is `/bin/sh`. In fact, much of the above discussion is applicable to `/bin/sh`. The next section will introduce many features particular to `csh`, so you should change your shell to `csh` before you begin reading it.

3. DETAILS ON THE SHELL FOR TERMINAL USERS

3.1 Shell Startup and Termination

When you log in, the shell is started by the system in your `home` directory and begins by reading commands from a file `.cshrc` in this directory. All shells which you may start during your terminal session will read from this file. You will later see what kinds of commands are usefully placed there. For now you do not need to have this file, and the shell will not complain about its absence.

A *login shell*, executed after you log in to the system, will, after it reads commands from `.cshrc`, read commands from a file `.login`, also in your home directory. This file contains commands which you wish to do each time you log in to the UNIX System. A `.login` file looks something like:

```
set ignoreeof
set mail=(/usr/mail/bill)
alias ts \
        'set noglob ;\
        eval `tset −s −m dialup:c100rv4pna −m plugboard:?hp2621nl *`';
ts; stty intr ^C kill ^U crt
set time=15 history=10
if (−e   $mail) then
        echo "${prompt}mail"
        mail
endif
```

This file contains several commands to be executed by the UNIX System each time a user logs in. The first is a *set* command which is interpreted directly by the shell. It sets the shell variable *ignoreeof* which causes the shell to not log off a user who types ^D. Rather, the user uses the *logout* command to log off of the system. By setting the *mail* variable, the user asks the shell to watch for incoming mail for the user. Every 5 minutes the shell looks for this file and tells the user if more mail has arrived there.

Next, the shell variable *time* was set to '15,' causing the shell to automatically print out statistics lines for commands that execute for at least 15 seconds of CPU time. The variable *history* is set to '10,' indicating that the user wants the shell to remember the last 10 commands typed in its *history list* (described later).

The user then created an *alias* "ts" which executes a *tset*(1) command setting up the modes of the terminal. The parameters to `tset` indicate the kinds of terminals that the user usually uses when not on a hardwired port. Then "ts" is executed and the `stty` command is also used to change the interrupt character to ^C and the line kill character to ^U.

Finally, if the user's mailbox file exists, then the user runs the `mail` program to process the mail.

When the `mail` program finishes, the shell will finish processing the user's `.login` file and begin reading commands from the terminal, prompting for each with %. When the user logs off (by giving the *logout* command), the shell will print 'logout' and execute commands from the file `.logout` if it exists in the user's home directory. After that the shell will terminate, and the UNIX System will log the user off the system. If the system is not being shut down, the user will receive a new login message. In any case, after the 'logout' message the shell is committed to terminating and will take no further input from the user's terminal.

3.2 Shell Variables

The shell maintains a set of *variables.* You saw the variables *history* and *time*, which had the values '10' and '15,' above. In fact, each shell variable has as its value an array of zero or more *strings.* Shell variables may be assigned values by the *set* command. It has several forms, the most useful of which was given above and is:

```
set name=value
```

Shell variables may be used to store values which are to be used in commands later through a substitution mechanism. The shell variables most commonly referenced are, however, those which the shell itself refers to. By changing the values of these variables one can directly affect the behavior of the shell.

One of the most important variables is the variable *path.* This variable contains a sequence of directory names where the shell searches for commands. The *set* command with no arguments shows the value of all variables currently defined (*set*) in the shell. The default value for path will be shown by *set* to be:

```
% set
argv      ()
cwd       /usr/bill
home      /usr/bill
path      (. /usr/ucb /bin /usr/bin)
prompt    %
shell     /bin/csh
status    0
term      c100rv4pna
user      bill
%
```

This output indicates that the variable path points to the current directory (`.`), and then `/usr/ucb`, `/bin`, and `/usr/bin`. Commands which you may write might be in **.** (usually one of your directories). Commands developed at Berkeley live in `/usr/ucb`, while commands developed at USL (formerly AT&T Bell Laboratories) live in `/bin` and `/usr/bin`.

A number of locally developed programs on the system live in the directory `/usr/local/bin`. If all shells that are invoked should have access to these new programs, place the command:

```
set path=(. /usr/ucb /bin /usr/bin /usr/local/bin)
```

in the file `.cshrc` in the user's home directory. Try doing this, and then log out and log back in and type:

```
set
```

again to see that the value assigned to *path* has changed.

One thing you should be aware of is that the shell examines each directory that you insert into your path and determines which commands are contained there. Except for the current directory `.`, which the shell treats specially, this means that if commands are added to a directory in your search path after you have started the shell, they will not necessarily be found by the shell. If you wish to use a command which has been added in this way, you should give the command:

```
rehash
```

to the shell, which will cause it to recompute its internal table of command locations, so that it will find the newly added command. Since the shell has to look in the current directory `.` on each command, placing it at the end of the path specification usually works equivalently and reduces overhead.

Other useful builtin variables are the variable *home* which shows your home directory, *cwd* which contains your current working directory, and the variable *ignoreeof* which can be set in your `.login` file to tell the shell not to exit when it receives an end-of-file from a terminal (as described above). The variable *ignoreeof* is one of several variables which the shell does not care about the value of, only whether they are *set* or *unset*. Thus, to set this variable you simply do:

```
set ignoreeof
```

and to unset it do:

```
unset ignoreeof
```

These give the variable *ignoreeof* no value, but none is desired or required.

Finally, some other builtin shell variables of use are the variables *noclobber* and *mail*. The metasyntax:

```
> filename
```

which redirects the standard output of a command will overwrite and destroy the previous contents of the named file. In this way you may accidentally overwrite a file which is valuable. If you would prefer that the shell not overwrite files in this way, you can:

```
set noclobber
```

in your `.login` file. Then trying to do:

```
date > now
```

would cause a diagnostic if `now` existed already. You could type:

```
date >! now
```

if you really wanted to overwrite the contents of now. The '>!' is a special metasyntax indicating that clobbering the file is all right.[4]

3.3 The Shell's History List

The shell can maintain a *history list* into which it places the words of previous commands. It is possible to use a notation to reuse commands or words from commands in forming new commands. This mechanism can be used to repeat previous commands or to correct minor typing mistakes in commands.

The following figure gives a sample session involving typical usage of the history mechanism of the shell:

4. The space between the ! and the word 'now' is critical here, as '!now' would be an invocation of the *history* mechanism, and have a totally different effect.

```
% cat bug.c
main()

{
        printf("hello);
}
% cc !$
cc bug.c
"bug.c", line 4: newline in string or char constant
"bug.c", line 5: syntax error
% ed !$
ed bug.c
29
4s/);/"&/p
        printf("hello");
w
30
q
% !c
cc bug.c
% a.out
hello% !e
ed bug.c
30
4s/lo/lo\\n/p
        printf("hello\n");
w
32
q
% !c −o bug
cc bug.c −o bug
% size a.out bug
a.out: 2784+364+1028 = 4176b = 0x1050b
bug: 2784+364+1028 = 4176b = 0x1050b
% ls −l !*
ls −l a.out bug
−rwxr−xr−x 1 bill      3932 Dec 19 09:41 a.out
−rwxr−xr−x 1 bill      3932 Dec 19 09:42 bug
% bug
hello
% pr -n -t bug.c | grpp -v '[0-9].$'
grpp: Command not found.
```

```
% ^grpp^grep
pr -n -t bug.c | grep -v '[0-9].$'
        1  main()
        3  {
        4        printf("hello\n");
        5  }
% !! | lpr
pr -n -t bug.c | grep -v '[0-9].$' | lpr
%
```

This example illustrates a very simple C program which has a bug (or two) in it in the file `bug.c`, which was `cat`ted out on the terminal. Then the C compiler was run on it. The file was referred to again as '!$' which meant the last argument to the previous command. Here the '!' is the history mechanism invocation metacharacter, and the '$' stands for the last argument, by analogy to '$' in the editor which stands for the end of the line. The shell echoed the command, as it would have been typed without use of the history mechanism, and then executed it. The compilation yielded error diagnostics so the user ran the editor on the file being compiled, fixed the bug, and ran the C compiler again, this time referring to this command simply as '!c,' which repeats the last command which started with the letter 'c.' If there were other commands starting with 'c' done recently, the user could have said '!cc' or even '!cc:p' which would have printed the last command starting with 'cc' without executing it.

After this recompilation, the user ran the resulting `a.out` file, noted that there still was a bug, and ran the editor again. After fixing the program, the user ran the C compiler again but tacked onto the command an extra `-o bug` that told the compiler to place the resultant binary in the file `bug` rather than `a.out`. In general, the history mechanisms can be used anywhere in the formation of new commands, and other characters can be placed before and after the substituted commands.

The user then ran the `size` command to see how large the binary program images created were, and then an `ls -l` command with the same argument list, denoting the argument list '*.' Finally, the user ran the program `bug` to see that its output was indeed correct.

To make a numbered listing of the program, the `pr` command was run on the file `bug.c`. In order to compress out blank lines in the output of `pr`, the output was run through the filter 'grep,' but it was misspelled as grpp. To correct this, a shell substitute was used that placed the old text and new text between '^' characters. This is similar to the substitute command in the editor. Finally, the same command was repeated with '!!,' but its output was sent to the line printer.

There are other mechanisms available for repeating commands. The *history* command prints out a number of previous commands with numbers by which they can be referenced. There is a way to refer to a previous command by searching for a string which appeared in it, and there are other, less useful, ways to select arguments to include in a new command. A complete description of all these mechanisms is given in *csh*(1) in the *INTERACTIVE UNIX System User's/System Administrator's Reference Manual*.

3.4 Aliases

The shell has an *alias* mechanism which can be used to make transformations on input commands. This mechanism can be used to simplify the commands you type, to supply default arguments to commands, or to perform transformations on commands and their arguments. The alias facility is similar to a macro facility. Some of the features obtained by aliasing can be obtained also using shell command files, but these take place in another instance of the shell and cannot directly affect the current shell's environment or involve commands such as `cd` which must be done in the current shell.

As an example, suppose that there is a new version of the `mail` program on the system called `newmail` you wish to use, rather than the standard `mail` program which is called `mail`. If you place the shell command:

```
alias mail newmail
```

in your `.cshrc` file, the shell will transform an input line of the form:

```
mail bill
```

into a call on `newmail`. More generally, suppose you wish the command `ls` to always show sizes of files, that is to always do `-s`. You can do:

```
alias ls ls −s
```

or even:

```
alias dir ls −s
```

creating a new command syntax ‘dir’ which does an `ls -s`. If you say:

```
dir ˜bill
```

then the shell will translate this to:

```
ls −s /usr/bill
```

Thus the *alias* mechanism can be used to provide short names for commands, to provide default arguments, and to define new short commands in terms of other commands. It is also possible to define aliases which contain multiple commands or pipelines, showing where the arguments to the original command are to be substituted using the facilities of the history mechanism. Thus the definition:

```
alias cd ′cd \!* ; ls ′
```

would do an `ls` command after each change directory `cd` command. The entire alias definition was enclosed in ′ characters to prevent most substitutions from occurring and the character semicolon (;) from being recognized as a metacharacter. The ! here is escaped with a \ to prevent it from being interpreted when the alias command is typed in. The ‘\!*’ here substitutes the entire argument list to the pre-aliasing `cd` command, without giving an error if there were no arguments. The ; separating commands is used here to indicate that one command should be done first and then the next one should be done. Similarly the definition:

```
alias whois ′grep \!^ /etc/passwd′
```

defines a command which looks up its first argument in the password file.

☛ Note that the shell reads the `.cshrc` file each time it starts up. If you place a large number of commands there, shells will tend to start slowly. You should try to limit the number of aliases you have to a reasonable number − 10 or 15 is reasonable; 50 or 60 may cause a noticeable delay in starting up shells and make the system seem sluggish when you execute commands from within the editor and other programs.

3.5 More Redirection; >> and >&

There are a few more notations useful to the terminal user that have not been introduced yet.

In addition to the standard output, commands also have a *diagnostic output* which is normally directed to the terminal even when the standard output is redirected to a file or a pipe. It is occasionally desirable to direct the diagnostic output along with the standard output. For instance, if you want to redirect the output of a long-running command into a file and wish to have a record of any error diagnostic it produces, you can do:

```
command >& file
```

The '>&' here tells the shell to route both the diagnostic output and the standard output into `file`. Similarly you can give the command:

```
command |& lpr
```

to route both standard and diagnostic output through the pipe to the line printer daemon *lpr*.[5]

Finally, it is possible to use the form:

```
command >> file
```

to place output at the end of an existing file.[6]

3.6 Jobs; Background, Foreground, or Suspended

When one or more commands are typed together as a pipeline or as a sequence of commands separated by semicolons, a single *job* is created by the shell consisting of these commands together as a unit. Single commands without pipes or semicolons create the simplest jobs. Usually, every line typed to the shell creates a job. Some lines that create jobs (one per line) are:

5. A command of the form:

```
command >&! file
```

exists, and is used when *noclobber* is set and `file` already exists.

6. If *noclobber* is set, then an error will result if `file` does not exist; otherwise the shell will create `file` if it does not exist. A form:

```
command >>! file
```

makes it not be an error for `file` to not exist when *noclobber* is set.

```
sort < data
ls -s | sort -n | sed 5q
mail harold
```

If the metacharacter ampersand (&) is typed at the end of the commands, then the job is started as a *background* job. This means that the shell does not wait for it to complete but immediately prompts and is ready for another command. The job runs *in the background* at the same time that normal jobs, called *foreground* jobs, continue to be read and executed by the shell one at a time. Thus:

```
du > usage &
```

would run the `du` program, which reports on the disk usage of your working directory (as well as any directories below it), put the output into the file `usage` and return immediately with a prompt for the next command without waiting for `du` to finish. The `du` program would continue executing in the background until it finished, even though you can type and execute more commands in the meantime. When a background job terminates, a message is typed by the shell just before the next prompt telling you that the job has completed. In the following example the `du` job finishes sometime during the execution of the `mail` command and its completion is reported just before the prompt after the `mail` job is finished:

```
% du > usage &
[1] 503
% mail bill
How do you know when a background job is finished?
EOT
[1] - Done          du > usage
%
```

If the job did not terminate normally, the 'Done' message might say something else like 'Killed.' If you want the terminations of background jobs to be reported at the time they occur (possibly interrupting the output of other foreground jobs), you can set the *notify* variable. In the previous example this would mean that the 'Done' message might have come right in the middle of the message to Bill. Background jobs are unaffected by any signals from the keyboard like the STOP, INTERRUPT, or QUIT signals mentioned earlier.

Jobs are recorded in a table inside the shell until they terminate. In this table, the shell remembers the command names, arguments, and the *process numbers* of all commands in the job as well as the working directory where the job was started. Each job in the table is either running *in the foreground* with the shell waiting for it to terminate, running *in the background,* or *suspended.* Only one job can be running in the foreground at one time, but several jobs can be suspended or running in the background at once. As each job is started, it is assigned a small identifying number called the *job number* which can be used later to refer to the job in the commands described below. Job numbers remain the same until the job terminates and then are re-used.

When a job is started in the backgound using '&,' its number, as well as the process numbers of all its (top level) commands, is typed by the shell before prompting you for another command. For example:

```
% ls –s | sort –n > usage &
[2] 2034 2035
%
```

runs the `ls` program with the `-s` option, and pipes this output into the `sort` program with the `-n` option which puts its output into the file `usage`. Since the '&' was at the end of the line, these two programs were started together as a background job. After starting the job, the shell prints the job number in brackets (two in this case) followed by the process number of each program started in the job. Then the shell immediately prompts for a new command, leaving the job running simultaneously.

The job control system permits the user control over groups of jobs after they have been started. A job may be placed in the background, brought to the foreground, stopped, restarted, or killed. Job control consists of extensions to the kernel signal processing and terminal interface, as well as extensions to the `csh` login shell. The "switch" character in the terminal driver can be used to signal the login shell that the current foreground process should be stopped. From there, the shell takes control as usual and accepts commands to start jobs, place them into the background, or kill them. The Bourne shell, `sh`, does not have job control features. By default, the switch keystroke used to suspend a job is undefined (job control is disabled). To define the job control switch character (enable job control), use the `stty` command, for example:

```
stty swtch ^z
```

Hereafter, the examples in this section assume job control has been enabled in this manner.

As mentioned above and in section 2.8, foreground jobs become *suspended* by typing ^Z which sends a STOP signal to the currently running foreground job. A background job can become suspended by using the *stop* command described below. When jobs are suspended they merely stop any further progress until started again, either in the foreground or the backgound. The shell notices when a job becomes stopped and reports this fact, much like it reports the termination of background jobs. For foreground jobs this looks like:

```
% du > usage
^Z
Stopped
%
```

The 'Stopped' message is typed by the shell when it notices that the du program stopped. For background jobs, using the *stop* command, it is:

```
% sort usage &
[1] 2345
% stop %1
[1] + Stopped (signal)      sort usage
%
```

Suspending foreground jobs can be very useful when you need to temporarily change what you are doing (execute other commands) and then return to the suspended job. Also, foreground jobs can be suspended and then continued as background jobs using the *bg* command, allowing you to continue other work and stop waiting for the foreground job to finish. Thus:

```
% du > usage
^Z
Stopped
% bg
[1] du > usage &
%
```

starts du in the foreground, stops it before it finishes, then continues it in the background allowing more foreground commands to be executed. This is especially helpful when a foreground job ends up taking longer than you expected and you wish you had started it in the backgound in the beginning.

All *job control* commands can take an argument that identifies a particular job. All job name arguments begin with the character %, since some of the job control commands also accept process numbers (printed by the `ps` command.) The default job (when no argument is given) is called the *current* job and is identified by a plus sign (+) in the output of the *jobs* command, which shows you which jobs you have. When only one job is stopped or running in the background (the usual case), it is always the current job; thus no argument is needed. If a job is stopped while running in the foreground, it becomes the *current* job and the existing current job becomes the *previous* job – identified by a '–' in the output of *jobs*. When the current job terminates, the previous job becomes the current job. When given, the argument is either '%–' (indicating the previous job); '%#' (where # is the job number); '%pref' (where *pref* is some unique prefix of the command name and arguments of one of the jobs); or '%?' followed by some string found in only one of the jobs.

The *jobs* command types the table of jobs, giving the job number, commands, and status ('Stopped' or 'Running') of each backgound or suspended job. With the `-l` option the process numbers are also typed:

```
% du > usage &
[1] 3398
% ls –s | sort –n > myfile &
[2] 3405
% mail bill
^Z
Stopped
% jobs
[1] – Running     du > usage
[2] Running       ls –s | sort –n > myfile
[3] + Stopped     mail bill
% fg %ls
ls –s | sort –n > myfile
% more myfile
```

The *fg* command is used to restart a previously suspended job[7] or change a background job to run in the foreground (allowing signals or input from the terminal). In the above example *fg* was used to change the `ls` job from the background to the foreground since the user wanted to wait for it to finish before looking at its output file. The *bg* command runs a suspended job in the background. It is usually used after stopping the currently running foreground job with the STOP signal. The combination of the STOP signal and the *bg* command changes a foreground job into a background job. The *stop* command suspends a background job.

The `kill` command terminates a background or suspended job immediately. In addition to jobs, it may be given process numbers as arguments, as printed by `ps`. Thus, in the example above, the running `du` command could have been terminated by the command:

```
% kill %1
[1] Terminated    du > usage
%
```

The *notify* command (not the variable mentioned earlier) indicates that the termination of a specific job should be reported at the time it finishes instead of waiting for the next prompt.

If a job running in the background tries to read input from the terminal, it is automatically stopped. When such a job is then run in the foreground, input can be given to the job. If desired, the job can be run in the background again until it requests input again. This is illustrated in the following sequence where the *s* command in the text editor might take a long time.

7. While all INTERACTIVE UNIX System binaries work with the job control signals, certain user-level utilities, such as the `vi` and TEN/PLUS* editors, have been modified to allow better use of job control. For example, if the `vi` editor is stopped and then restarted, the screen is repainted.

```
% ed bigfile
120000
1,$s/thisword/thatword/
^Z
Stopped
% bg
[1] ed bigfile &
%
 . . . some foreground commands
[1] Stopped (tty input) ed bigfile
% fg
ed bigfile
w
120000
q
%
```

So after the *s* command was issued, the `ed` job was stopped with ^Z and then put in the background using *bg*. Some time later when the *s* command was finished, `ed` tried to read another command and was stopped because jobs in the backgound cannot read from the terminal. The *fg* command returned the `ed` job to the foreground where it could once again accept commands from the terminal.

The command:

```
stty tostop
```

causes all background jobs run on your terminal to stop when they are about to write output to the terminal. This prevents messages from background jobs from interrupting foreground job output and allows you to run a job in the background without losing terminal output. It also can be used for interactive programs that sometimes have long periods without interaction. Thus, each time it outputs a prompt for more input, it will stop before the prompt. It can then be run in the foreground using *fg*, more input can be given, and, if necessary, stopped and returned to the background. This `stty` command might be a good thing to put in your `.login` file if you do not like output from background jobs interrupting your work. It also can reduce the need for redirecting the output of background jobs if the output is not very big:

```
% stty tostop
% wc hugefile &
[1] 10387
% ed text
. . . some time later
q
[1] Stopped (tty output) wc hugefile
% fg wc
wc hugefile
  13371  30123  302577
% stty −tostop
```

Thus after some time, the `wc` command, which counts the lines, words, and characters in a file, had one line of output. When it tried to write this to the terminal, it stopped. By restarting it in the foreground the user allowed it to write on the terminal exactly when he was ready to look at its output. Programs which attempt to change the mode of the terminal will also block, whether or not *tostop* is set, when they are not in the foreground, as it would be very unpleasant to have a background job change the state of the terminal.

Since the *jobs* command only prints jobs started in the currently executing shell, it knows nothing about background jobs started in other login sessions or within shell files. The `ps` can be used in this case to find out about background jobs not started in the current shell.

3.7 Working Directories

As mentioned in section 2.6, the shell is always in a particular *working directory*. The 'change directory' command, `chdir` (its short form `cd` may also be used), changes the working directory of the shell, that is, changes the directory you are located in.

It is useful to make a directory for each project you wish to work on and to place all files related to that project in that directory. The 'make directory' command, `mkdir`, creates a new directory. The `pwd` ('print working directory') command reports the absolute pathname of the working directory of the shell, that is, the directory you are located in. Thus in the example below:

```
% pwd
/usr/bill
% mkdir newpaper
% chdir newpaper
% pwd
/usr/bill/newpaper
%
```

the user has created and moved to the directory `newpaper`, where, for example, he might place a group of related files.

No matter where you have moved to in a directory hierarchy, you can return to your *home* login directory by doing just:

```
cd
```

with no arguments. The name '..' always means the directory above the current one in the hierarchy, thus:

```
cd ..
```

changes the shell's working directory to the one directly above the current one. The name '..' can be used in any path name, thus:

```
cd ../programs
```

means change to the directory `programs` contained in the directory above the current one. If you have several directories for different projects under, say, your home directory, this shorthand notation permits you to switch easily between them.

The shell always remembers the path name of its current working directory in the variable *cwd.* The shell can also be requested to remember the previous directory when you change to a new working directory. If the 'push directory' command *pushd* is used in place of the `cd` command, the shell saves the name of the current working directory on a *directory stack* before changing to the new one. You can see this list at any time by typing the 'directories' command *dirs:*

```
% pushd newpaper/references
% pushd /usr/lib/tmac
/usr/lib/tmac ~/newpaper/references ~
% dirs
/usr/lib/tmac ~/newpaper/references ~
% popd
% popd
%
```

The list is printed in a horizontal line, reading left to right, with a tilde (~) as shorthand for your home directory—in this case `/usr/bill`. The directory stack is printed whenever there is more than one entry on it and it changes. It is also printed by a *dirs* command. *dirs* is usually faster and more informative than `pwd` since it shows the current working directory as well as any other directories remembered in the stack.

The *pushd* command with no argument alternates the current directory with the first directory in the list. The 'pop directory' *popd* command without an argument returns you to the directory you were in prior to the current one, discarding the previous current directory from the stack (forgetting it). Typing *popd* several times in a series takes you backward through the directories you had been in (changed to) by the *pushd* command. There are other options to *pushd* and *popd* to manipulate the contents of the directory stack and to change to directories not at the top of the stack; see the *csh*(1) manual entry for details.

Since the shell remembers the working directory in which each job was started, it warns you when you might be confused by restarting a job in the foreground which has a different working directory than the current working directory of the shell. Thus if you start a background job, then change the shell's working directory and then cause the background job to run in the foreground, the shell warns you that the working directory of the currently running foreground job is different from that of the shell.

```
% dirs −l
/usr/bill
% cd myproject
% dirs
% ed prog.c
1143
^Z
Stopped
% cd ..
% ls
myproject
textfile
% fg
ed prog.c (wd: ~/myproject)
```

This way the shell warns you when there is an implied change of working directory, even though no `cd` command was issued. In the above example, the `ed` job was still in `/usr/bill/project` even though the shell had changed to `/usr/bill`. A similar warning is given when such a foreground job terminates or is suspended (using the STOP signal) since the return to the shell again implies a change of working directory.

```
% fg
ed prog.c (wd: ~/myproject)
 . . . after some editing
q
(wd now: ~)
%
```

These messages are sometimes confusing if you use programs that change their own working directories, since the shell only remembers which directory a job is started in, and assumes it stays there. The `-l` option of *jobs* will type the working directory of suspended or background jobs when it is different from the current working directory of the shell.

3.8 Useful Builtin Commands

This section discusses a few of the useful builtin commands of the shell and describes how they are used.

The *alias* command described above is used to assign new aliases and to show the existing aliases. With no arguments it prints the current aliases. It may also be given only one argument, such as:

```
alias ls
```

to show the current alias for `ls`, for example.

The `echo` command prints its arguments. It is often used in *shell scripts* or as an interactive command to see what file name expansions will produce.

The *history* command will show the contents of the history list. The numbers given with the history events can be used to reference previous events which are difficult to reference using the contextual mechanisms introduced above. There is also a shell variable called *prompt.* By placing a ! character in its value, the shell will there substitute the number of the current command in the history list. You can use this number to refer to this command in a history substitution. Thus you could:

```
set prompt='\! % '
```

Note that the '!' character had to be *escaped* here even within '' characters.

The `limit` command is used to restrict use of resources. When used without arguments, it prints the current limitations (actual numbers may vary):

```
filesize        12880 blocks
datasize        20376 kbytes
```

Limits can be set, for example:

```
limit filesize 4096
```

Most reasonable units abbreviations will work; see the *csh*(1) manual entry for more details.

The *logout* command can be used to terminate a login shell which has *ignoreeof* set.

The *rehash* command causes the shell to recompute a table of where commands are located. This is necessary if you add a command to a directory in the current shell's search path and wish the shell to find it, since otherwise the hashing algorithm may tell the shell that the command was not in that directory when the hash table was computed.

The *repeat* command can be used to repeat a command several times. Thus to make five copies of the file *one* in the file *five*, you could do:

```
repeat 5 cat one >> five
```

The *setenv* command can be used to set variables in the environment. Thus:

```
setenv TERM adm3a
```

will set the value of the environment variable TERM to 'adm3a'. The *setenv* command will print out the environment if used without arguments:

```
% setenv
HOME=/usr/bill
SHELL=/bin/csh
PATH=:/usr/ucb:/bin:/usr/bin:/usr/local
TERM=adm3a
USER=bill
%
```

The *source* command can be used to force the current shell to read commands from a file. Thus:

```
source .cshrc
```

can be used after editing in a change to the `.cshrc` file which you wish to take effect right away.

The `time` command can be used to cause a command to be timed no matter how much CPU time it takes. Thus:

```
% time cp /etc/inittab /usr/bill/inittab
0.0u 0.1s 0:01 15%
% time wc /etc/inittab /usr/bill/inittab
    46   195   2083 /etc/inittab
    46   195   2083 /usr/bill/inittab
    92   390   4166 total
0.1u 0.1s 0:00 15%
%
```

indicates that the `cp` command used a negligible amount of user time (u) and about 1/10th of a system time (s), and the elapsed time was 1 second (0:01). The word count command `wc`, on the other hand, used 0.1 seconds of user time and 0.1 seconds of system time in less than 1 second of elapsed ("wall clock") time. The percentage "15%" indicates that over the period when it was active, the command wc used an average of 15 percent of the available CPU cycles of the machine.

The *unalias* and *unset* commands can be used to remove aliases and variable definitions from the shell, and *unsetenv* removes variables from the environment.

3.9 What Else?

This concludes the basic discussion of the shell for terminal users. There are more features of the shell to be discussed here, and all features of the shell are discussed in its manual entries. One useful feature which is discussed later is the *foreach* builtin command which can be used to run the same command sequence with a number of different arguments.

If you intend to use the UNIX System a lot, you should look through the rest of this document and the *csh*(1) manual entry to become familiar with the other facilities which are available to you.

4. SHELL CONTROL STRUCTURES AND COMMAND SCRIPTS

4.1 Introduction

It is possible to place commands in files and to cause shells to be invoked to read and execute commands from these files, which are called *shell scripts*. Detailed here are those features of the shell useful to the writers of such scripts.

4.2 Make

It is important to first note what shell scripts are *not* useful for. There is a program called `make` which is very useful for maintaining a group of related files or performing sets of operations on related files. For instance, a large program consisting of one or more files can have its dependencies described in a `makefile` which contains definitions of the commands used to create these different files when changes occur. Definitions of the means for printing listings, cleaning up the directory in which the files reside, and installing the resultant programs are easily and most appropriately placed in this `makefile`. This format is superior and preferable to maintaining a group of shell procedures to maintain these files.

Similarly, when working on a document, a `makefile` may be created which defines how different versions of the document are to be created and which options of `nroff` or `troff` are appropriate.

4.3 Invocation and the *Argv* Variable

A `csh` command script may be interpreted by saying:

```
% csh script ...
```

where `script` is the name of the file containing a group of `csh` commands and '...' is replaced by a sequence of arguments. The shell places these arguments in the variable *argv* and then begins to read commands from the script. These parameters are then available through the same mechanisms which are used to reference any other shell variables.

If you make the file `script` executable by doing:

```
chmod 755 script
```

and place a shell comment at the beginning of the shell script (i.e., begin the file with `#!/bin/csh`), then a `/bin/csh` will automatically be invoked to execute `script` when you type:

```
script
```

If the file does not begin with `#!/bin/csh`, then the standard shell `/bin/sh` will be used to execute it. This allows you to convert your older shell scripts to use `csh` at your convenience.

4.4 Variable Substitution

After each input line is broken into words and history substitutions are done on it, the input line is parsed into distinct commands. Before each command is executed, a mechanism known as *variable substitution* is done on these words. Keyed by the character dollar sign ($), this substitution replaces the names of variables by their values. Thus:

```
echo $argv
```

when placed in a command script would cause the current value of the variable *argv* to be echoed to the output of the shell script. It is an error for *argv* to be unset at this point.

A number of notations are provided for accessing components and attributes of variables. The notation:

```
$?name
```

expands to '1' if name is *set* or to '0' if name is not *set*. It is the fundamental mechanism used for checking whether particular variables have been assigned values. All other forms of reference to undefined variables cause errors.

The notation:

```
$#name
```

expands to the number of elements in the variable *name*. Thus:

```
% set argv=(a b c)
% echo $?argv
1
% echo $#argv
3
% unset argv
% echo $?argv
0
% echo $argv
Undefined variable: argv.
%
```

It is also possible to access the components of a variable which has several values. Thus:

```
$argv[1]
```

gives the first component of *argv* or, in the example above, 'a.' Similarly:

```
$argv[$#argv]
```

would give 'c,' and:

```
$argv[1-2]
```

would give 'a b.' Other notations useful in shell scripts are:

$*n*

where *n* is an integer as a shorthand for:

$argv[*n*]

the *nth* parameter and:

```
$*
```

which is a shorthand for:

```
$argv
```

The form:

```
$$
```

expands to the process number of the current shell. Since this process number is unique in the system, it can be used in the generation of unique temporary file names. The form:

```
$<
```

is quite special and is replaced by the next line of input read from the shell's standard input (not the script it is reading). This is useful for writing shell scripts that are interactive, reading commands from the terminal, or even writing a shell script that acts as a filter, reading lines from its input file. Thus, the sequence:

```
echo -n 'yes or no?'
set a=($<)
```

would write out the prompt 'yes or no?' without a new-line character and then read the answer into the variable 'a.' In this case, '$#a' would be '0' if either a blank line or end-of-file (^D) was typed.

One minor difference between '$*n*' and '$argv[*n*]' should be noted here. The form '$argv[*n*]' will yield an error if *n* is not in the range '1−$#argv,' while '$n' will never yield an out of range subscript error. This is for compatibility with the way older shells handled parameters.

Another important point is that it is never an error to give a subrange of the form 'n−'; if there are less than *n* components of the given variable, then no words are substituted. A range of the form 'm-n' likewise returns an empty vector without giving an error when *m* exceeds the number of elements of the given variable, provided the subscript *n* is in range.

4.5 Expressions

In order for interesting shell scripts to be constructed, it must be possible to evaluate expressions in the shell based on the values of variables. In fact, all the arithmetic operations of the language C are available in the shell with the same precedence that they have in C. In particular, the operations '==' and '!=' compare strings and the operators '&&' and '||' implement the boolean and/or operations. The special operators '=~' and '!~' are similar to '==' and '!=' except that the string on the right side can have pattern matching characters (such as *, ?, or []), and the test is whether the string on the left matches the pattern on the right.

The shell also allows file enquiries of the form:

```
−? filename
```

where ? is replaced by a number of single characters. For instance, the expression primitive:

    −e filename

tells whether the file `filename` exists. Other primitives test for read, write, and execute access to the file, whether it is a directory, or has nonzero length.

It is possible to test whether a command terminates normally, by a primitive of the form '{ command }' which returns true, i.e., '1' if the command succeeds exiting normally with exit status 0, or '0' if the command terminates abnormally or with exit status nonzero. If more detailed information about the execution status of a command is required, it can be executed and the variable '$status' examined in the next command. Since '$status' is set by every command, it is very transient. It can be saved if it is inconvenient to use it only in the single immediately following command.

For a full list of expression components available, see *csh*(1).

4.6 Sample Shell Script

A sample shell script which makes use of the expression mechanism of the shell and some of its control structure follows:

```
% cat copyc
#!/bin/csh
# Copyc copies those C programs in the specified list
# to the directory ~/backup if they differ from the files
# already in ~/backup
#
set noglob
foreach i ($argv)

        if ($i !~ *.c) continue  # not a .c file so do nothing

        if (! -r ~/backup/$i:t) then
              echo $i:t not in backup... not cp\'ed
              continue
        endif

        cmp -s $i ~/backup/$i:t # to set $status

        if ($status != 0) then
              echo new backup of $i
              cp $i ~/backup/$i:t
        endif
end
```

This script makes use of the *foreach* builtin command, which causes the shell to execute the commands between the *foreach* and the matching *end* for each of the values given between (and) with the named variable, in this case 'i' set to successive values in the list. Within this loop you may use the command *break* to stop executing the loop and *continue* to prematurely terminate one iteration and begin the next. After the *foreach* loop the iteration variable (*i* in this case) has the value at the last iteration.

Set the variable *noglob* here to prevent file name expansion of the members of *argv*. This is a good idea, in general, if the arguments to a shell script are file names which have already been expanded or if the arguments may contain file name expansion metacharacters. It is also possible to quote each use of a $ variable expansion, but this is harder and less reliable.

The other control construct used here is a statement of the form:

```
if ( expression ) then
        command
        . . .
endif
```

The placement of the keywords here is *not* flexible due to the current implementation of the shell.[8]

The shell does have another form of the *if* statement of the form:

```
if ( expression ) command
```

which can be written:

```
if ( expression ) \
                        command
```

Here the new-line character has been escaped for the sake of appearance. The command must not involve '|,' '&,' or ';' and must not be another control command. The second form requires the final '\' to *immediately* precede the end-of-line.

The more general *if* statements above also admit a sequence of *else–if* pairs followed by a single *else* and an *endif*, for example:

8. The following two formats are not currently acceptable to the shell:

```
if ( expression )       # Won't work!
then
                        command
                        ...
endif
```

and

```
if ( expression ) then command endif        # Won't work
if ( expression ) then
        commands
else if (expression ) then
        commands
...

else
        commands
endif
```

Another important mechanism used in shell scripts is the colon (:) modifier. You can use the modifier ':r' here to extract a root of a file name or ':e' to extract the *extension*. Thus if the variable *i* has the value '/mnt/foo.bar', then:

```
% echo $i $i:r $i:e
/mnt/foo.bar /mnt/foo bar
%
```

shows how the ':r' modifier strips off the trailing '.bar' and the the ':e' modifier leaves only the 'bar.' Other modifiers will take off the last component of a path name leaving the head ':h' or all but the last component of a path name leaving the tail ':t.' These modifiers are fully described in *csh*(1) and in this article. It is also possible to use the *command substitution* mechanism described in the next major section to perform modifications on strings to then reenter the shell's environment. Since each usage of this mechanism involves the creation of a new process, it is much more expensive to use than the ':' modification mechanism.[9] Finally, note that the character # lexically introduces a shell comment in shell scripts (but not from the terminal). All subsequent characters on the input line after a #

9. It is also important to note that the current implementation of the shell limits the number of ':' modifiers on a '$' substitution to 1. Thus:

```
% echo $i $i:h:t
/a/b/c /a/b:t
%
```

does not do what one would expect.

are discarded by the shell. This character can be quoted using ′ or \ to place it in an argument word.

4.7 Other Control Structures

The shell also has control structures *while* and *switch*, which are similar to those of C. These take the forms:

```
while ( expression )
        commands
end
```

and:

```
switch ( word )

case str1:
        commands
        breaksw

 ...

case strn:
        commands
        breaksw

default:
        commands
        breaksw

endsw
```

For details, see *csh*(1). C programmers should note that *breaksw* is used to exit from a *switch*, while *break* exits a *while* or *foreach* loop. A common mistake to make in csh scripts is to use *break* rather than *breaksw* in switches.

Finally, csh allows a *goto* statement, with labels looking like they do in C, that is:

```
loop:
        commands
        goto loop
```

4.8 Supplying Input to Commands

Commands run from shell scripts receive by default the standard input of the shell which is running the script. This is different from previous shells running under the UNIX System. It allows shell scripts to fully participate in pipelines, but mandates extra notation for commands which are to take inline data.

Thus you need a metanotation for supplying inline data to commands in shell scripts. As an example, consider this script which runs the editor to delete leading blanks from the lines in each argument file:

```
% cat deblank
# deblank –– remove leading blanks
foreach i ($argv)
ed – $i << ′EOF′
1,$s/^[ ]*//
w
q
′EOF′
end
%
```

The notation '<< ′EOF′' means that the standard input for the ed command is to come from the text in the shell script file up to the next line consisting of exactly '′EOF′'. The fact that the 'EOF' is enclosed in ′ characters, i.e., quoted, causes the shell to not perform variable substitution on the intervening lines. In general, if any part of the word following the '<<' which the shell uses to terminate the text to be given to the command is quoted, then these substitutions will not be performed. In this case, since you used the form '1,$' in the editor script, you needed to insure that this $ was not variable substituted. You could also have insured this by preceding the $ here with a \, that is:

```
1,\$s/^[ ]*//
```

but quoting the 'EOF' terminator is a more reliable way of achieving the same thing.

4.9 Catching Interrupts

If the shell script creates temporary files, you may wish to catch interruptions of the shell script so that you can clean up these files. You can then do:

```
onintr label
```

where *label* is a label in our program. If an interrupt is received, the shell will do a 'goto label,' and you can remove the temporary files and then do an `exit` command (which is built in to the shell) to exit from the shell script. If you wish to exit with a nonzero status, you can do:

```
exit(1)
```

for example, to exit with status '1'.

4.10 What Else?

There are other features of the shell useful to writers of shell procedures. The *verbose* and *echo* options and the related `-v` and `-x` command line options can be used to help trace the actions of the shell. The `-n` option causes the shell only to read commands and not to execute them and may sometimes be of use.

One other thing to note is that `csh` will not execute shell scripts which do not begin with `#!/bin/csh`. Shell files that do not begin this way will be executed by `sh`, so by using `#!/bin/csh`, the file will be interpreted as "run with `csh`."

There is also another quotation mechanism using double quotes (" ") which allows only some of the expansion mechanisms discussed up to this point to occur on the quoted string and serves to make this string into a single word as quote (′) does.

5. OTHER, LESS COMMONLY USED, SHELL FEATURES

5.1 Loops at the Terminal; Variables as Vectors

It is occasionally useful to use the *foreach* control structure at the terminal to aid in performing a number of similar commands. For instance, there were at one point three shells in use on the Cory UNIX System at Cory Hall, `/bin/sh`, `/bin/nsh`, and `/bin/csh`. To count the number of persons using each shell, one could have issued the commands:

```
% grep −c 'csh$' /etc/passwd
27
% grep −c 'nsh$' /etc/passwd
128
% grep −c −v 'sh$' /etc/passwd
430
%
```

Since these commands are very similar, you can use *foreach* to do this more easily:

```
% foreach i ('sh$' 'csh$' '−v sh$')
? grep −c $i /etc/passwd
? end
27
128
430
%
```

Note here that the shell prompts for input with ? when reading the body of the loop.

Very useful with loops are variables which contain lists of file names or other words. You can, for example, do:

```
% set a=(`ls`)
% echo $a
csh.n csh.rm
% ls
csh.n
csh.rm
% echo $#a
2
%
```

The *set* command here gave the variable *a* a list of all the file names in the current directory as value. You can then iterate over these names to perform any chosen function.

The output of a command within ` characters is converted by the shell to a list of words. You can also place the ` quoted string within " characters to take each (non-empty) line as a component of the variable, preventing the lines from being split into words at blanks and tabs. A modifier ':x' exists which can be used later to expand each component of the variable into another variable, splitting it into separate words at embedded blanks and tabs.

5.2 Braces { ... } in Argument Expansion

Another form of file name expansion, alluded to before, involves the curley brace characters { and }. These characters specify that the contained strings (separated by commas (,)) are to be consecutively substituted into the containing characters and the results expanded left to right. Thus:

```
A{str1,str2,...strn}B
```

expands to:

```
Astr1B Astr2B ... AstrnB
```

This expansion occurs before the other file name expansions and may be applied recursively (i.e. nested). The results of each expanded string are sorted separately, left to right order being preserved. The resulting file names are not required to exist if no other expansion mechanisms are used. This means that this mechanism can be used to generate arguments which are not file names, but which have common parts.

A typical use of this would be:

```
mkdir ~/{hdrs,retrofit,csh}
```

to make subdirectories `hdrs`, `retrofit`, and `csh` in your home directory. This mechanism is most useful when the common prefix is longer than in this example:

```
chown root /usr/{ucb/{ex,edit},lib/{ex?.?*,how_ex}}
```

5.3 Command Substitution

A command enclosed in ` characters is replaced, just before file names are expanded, by the output from that command. Thus, it is possible to do:

```
    set pwd=`pwd`
```

to save the current directory in the variable pwd or to do:

```
    ex `grep −l TRACE *.c`
```

to run the editor ex, supplying as arguments those files whose names end in '.c' which have the string 'TRACE' in them.[10]

5.4 Other Details Not Covered Here

In particular circumstances it may be necessary to know the exact nature and order of different substitutions performed by the shell. The exact meaning of certain combinations of quotations is also occasionally important. These are detailed fully in *csh*(1).

The shell has a number of command line option flags mostly of use in writing UNIX System programs and debugging shell scripts. See *csh*(1).

10. Command expansion also occurs in input redirected with << and within " quotations. Refer to the shell manual section for full details.

GLOSSARY

This glossary lists the most important terms introduced in the introduction to the shell and gives references to sections of the shell document for further information about them. References of the form *pr*(1) indicate that the command `pr` is in Section 1 in the *INTERACTIVE UNIX System User's/System Administrator's Reference Manual*. You can look at an on-line copy of this manual entry by typing:

```
man 1 pr
```

References of the form (3.5) indicate that more information can be found in section 3.5 of this manual.

. Your current directory has the name '.' as well as the name printed by the command `pwd`; see also *dirs*. The current directory '.' is usually the first *component* of the search path contained in the variable *path*, thus commands which are in '.' are found first (3.2). The character '.' is also used in separating *components* of file names (2.6). The character '.' at the beginning of a *component* of a *path name* is treated specially and not matched by the *file name expansion* metacharacters '?', '*', and '[' ']' pairs (2.6).

.. Each directory has a file '..' in it which is a reference to its parent directory. After changing into the directory with `chdir`, that is:

```
chdir paper
```

you can return to the parent directory by doing:

```
chdir ..
```

The current directory is printed by `pwd` (3.7).

a.out Compilers which create executable images create them, by default, in the file `a.out` for historical reasons (3.3).

absolute path name
A *path name* which begins with a '/' is *absolute* since it specifies the *path* of directories from the beginning of the entire directory system – called the `root` directory. *Path names* which are not *absolute* are called *relative* (see definition of *relative path name*) (2.6).

alias An *alias* specifies a shorter or different name for a UNIX System command, or a transformation on a command to be performed in the shell. The shell has a command *alias* which establishes *aliases* and can print their current values. The command *unalias* is used to remove *aliases* (3.4).

argument
Commands in the UNIX System receive a list of *argument* words. Thus the command:

```
echo a b c
```

consists of the *command name* 'echo' and three *argument* words 'a,' 'b,' and 'c.' The set of *arguments* after the *command name* is said to be the *argument list* of the command (2.1).

argv The list of arguments to a command written in the shell language (a shell script or shell procedure) is stored in a variable called *argv* within the shell. This name is taken from the conventional name in the C programming language (4.4).

background
Commands started without waiting for them to complete are called *background* commands (3.6).

base A file name is sometimes thought of as consisting of a *base* part, before any period character (.), and an *extension* – the part after the '.'. See *file name* and *extension* (2.6) and basename (2).

bg The *bg* command causes a *suspended* job to continue execution in the *background* (3.6).

bin A directory containing binaries of programs and shell scripts to be executed is typically called a `bin` directory. The standard system `bin` directories are `/bin` containing the most heavily used commands and `/usr/bin` which contains most other user programs. Programs developed at UC Berkeley live in `/usr/ucb`, while locally written programs live in `/usr/local/bin`. You can place binaries in any directory. If you wish to execute them often, the name of the directories should be a *component* of the variable *path*.

break *break* is a builtin command used to exit from loops within the control structure of the shell (4.7).

breaksw
The *breaksw* builtin command is used to exit from a *switch* control structure, like a *break* exits from loops (4.7).

builtin A command executed directly by the shell is called a *builtin* command. Most commands in the UNIX System are not built into the shell, but rather exist as files in `bin` directories. These commands are accessible because the directories in which they reside are named in the *path* variable.

case A *case* command is used as a label in a *switch* statement in the shell's control structure, similar to that of the language C. Details are given in the shell documentation *csh*(1) (4.7).

cat The `cat` program catenates a list of specified files on the *standard output*. It is usually used to look at the contents of a single file on the terminal, to '`cat` a file' (2.8, 3.3).

cd The `cd` command is used to change the *working directory*. With no arguments, `cd` changes your *working directory* to be your `home` directory (3.4, 3.7).

chdir The `chdir` command is a synonym for `cd`. `cd` is usually used because it is easier to type.

cmp `cmp` is a program which compares files. It is usually used on binary files, or to see if two files are identical

(4.6). For comparing text files the program `diff`, described in *diff*(1) is used.

command

A function performed by the system, either by the shell (a builtin *command*) or by a program residing in a file in a directory within the UNIX System, is called a *command* (2.1).

command name

When a command is issued, it consists of a *command name*, which is the first word of the command, followed by arguments. The convention on the UNIX System is that the first word of a command names the function to be performed (2.1).

command substitution

The replacement of a command enclosed in '`' characters by the text output by that command is called *command substitution* (5.3).

component

A part of a *path name* between slash (/) characters is called a *component* of that *path name*. A variable which has multiple strings as value is said to have several *components*; each string is a *component* of the variable.

continue

A builtin command which causes execution of the enclosing *foreach* or *while* loop to cycle prematurely. Similar to the *continue* command in the programming language C (4.6).

control-

Certain special characters, called *control* characters, are produced by holding down the CTRL key on your terminal and simultaneously pressing another character, much like the SHIFT key is used to produce uppercase characters. Thus *control*-c is produced by holding down the CTRL key while pressing the 'c' key.

core dump

When a program terminates abnormally, the system places an image of its current state in a file named `core`. This *core dump* can be examined with the system debugger *adb*(1) or *sdb*(1) in order to determine what went wrong with the program (2.8). If the shell produces a message of the form:

Illegal instruction (core dumped)

(where 'Illegal instruction' is only one of several possible messages), you should report this to the author of the program or a system administrator, saving the `core` file.

cp The `cp` (copy) program is used to copy the contents of one file into another file. It is one of the most commonly used UNIX System commands (2.6).

csh The name of the shell program that this document describes.

.cshrc The file `.cshrc` in your `home` directory is read by each shell as it begins execution. It is usually used to change the setting of the variable *path* and to set *alias* parameters which are to take effect globally (3.1).

cwd The *cwd* variable in the shell holds the *absolute path name* of the current *working directory*. It is changed by the shell whenever your current *working directory* changes and should not be changed otherwise (3.2).

date The `date` command prints the current date and time (2.3).

debugging

Debugging is the process of correcting mistakes in programs and shell scripts. The shell has several options and variables which may be used to aid in shell *debugging* (5.4).

default:

The label *default:* is used within shell *switch* statements, as it is in the C language to label the code to be executed if none of the *case* labels matches the value switched on (4.7).

DELETE

The $\boxed{\text{DELETE}}$ or $\boxed{\text{RUBOUT}}$ key on the terminal normally causes an interrupt to be sent to the current job. Many users change the interrupt character to be ^C.

detached

A command that continues running in the *background* after you log out is said to be *detached.*

diagnostic

An error message produced by a program is often referred to as a *diagnostic*. Most error messages are not written to the *standard output*, since that is often directed away from the terminal (2.3, 2.5). Error messsages are instead written to the *diagnostic output* which may be directed away from the terminal, but usually is not. Thus *diagnostics* will usually appear on the terminal (3.5).

directory

A structure which contains files. At any time you are in one particular *directory* whose names can be displayed by the command `pwd`. The `chdir` command will change you to another *directory* and make the files in that *directory* visible. The *directory* that you are in when you first log in is your *home* directory (2.1, 3.7).

directory stack

The shell saves the names of previous *working directories* in the *directory stack* when you change your current *working directory* via the *pushd* command. The *directory stack* can be displayed by using the *dirs* command, which includes your current *working directory* as the first directory name on the left (3.7).

dirs The *dirs* command displays the shell's *directory stack* (3.7).

du The `du` command is a program (described in *du*(1)) which displays the number of disk blocks in all directories below and including your current *working directory* (3.6).

echo The `echo` command prints its arguments (2.6, 4.6).

else The *else* command is part of the 'if-then-else-endif' control command construct (4.6).

endif If an *if* statement is ended with the word *then*, all lines following the *if* up to a line starting with the word *endif* or *else* are executed if the condition between parentheses after the *if* is true (4.6).

EOF An *end-of-file* is generated by the terminal by a $\boxed{\text{CTRL}}$ $\boxed{\text{d}}$, and whenever a command reads to the end of a file which it has been given as input. Commands receiving input from a *pipe* receive an *end-of-file* when the command sending them input completes. Most commands terminate when they receive an *end-of-file*. The shell has an option to ignore *end-of-file* from a terminal input which may help you keep from logging out accidentally by typing too many $\boxed{\text{CTRL}}$ $\boxed{\text{d}}$'s (2.1, 2.8, 4.8).

escape A '\' character used to prevent the special meaning of a metacharacter is said to *escape* the character from its special meaning. Thus:

```
echo \*
```

will echo the character '*', while just:

```
echo *
```

will echo the names of the file in the current directory. In this example, \ *escape*s '*' (2.7). There is also a non-printing character called *escape*, usually labelled $\boxed{\text{ESC}}$ or $\boxed{\text{ALTMODE}}$ on terminal keyboards. Some older UNIX systems use this character to indicate that output is to be *suspended*. Most systems use $\boxed{\text{CTRL}}$ $\boxed{\text{s}}$ to stop the output and $\boxed{\text{CTRL}}$ $\boxed{\text{q}}$ to start it.

/etc/passwd
This file contains information about the accounts currently on the system. It consists of a line for each account with fields separated by colon (:) characters (2.8). You can look at this file by saying:

```
cat /etc/passwd
```

The commands `finger` and `grep` are often used to search for information in this file. See *finger*(1), *grep*(1), and *passwd*(1) for more details.

exit The *exit* command is used to force termination of a shell script, and it is built into the shell (4.9).

exit status
A command which discovers a problem may reflect this back to the command (such as a shell) which invoked (executed) it. It does this by returning a nonzero number as its *exit status*, a status of zero being considered 'normal termination'. The *exit* command can be used to force a shell command script to give a nonzero *exit status* (4.6).

expansion
The replacement of strings in the shell input which contain metacharacters by other strings is referred to as the process of *expansion*. Thus the replacement of the character * by a sorted list of files in the current directory is a 'file name expansion'. Similarly the replacement of the characters !! by the text of the last command is a 'history expansion.' *Expansions* are also referred to as *substitutions* (2.6, 4.4, 5.2).

expressions
Expressions are used in the shell to control the conditional structures used in the writing of shell scripts and in calculating values for these scripts. The operators available in shell *expressions* are those of the language C (4.5).

extension
File names often consist of a *base* name and an *extension* separated by the character '.'. By convention, groups of related files often share the same `root` name. Thus if `prog.c` were a C program, then the object file for this program would be stored in `prog.o`. Similarly, a paper written with the `-me` nroff macro package might be stored in `paper.me`, while a formatted version of this paper might be kept in `paper.out` and a list of spelling errors in `paper.errs` (2.6).

fg The *job control* command *fg* is used to run a *background* or *suspended* job in the *foreground* (2.8, 3.6).

file name
Each file in the UNIX System has a name consisting of up to 14 characters, not including the character slash (/), which is used in *path name* building. Most *file names* do not begin with the character period (.); they contain only letters and digits with perhaps a . separating the *base* portion of the *file name* from an *extension* (2.6).

file name expansion
File name expansion uses the metacharacters *, ?, and [and] to provide a convenient mechanism for naming files. Using *file name expansion* it is easy to name all the files in the current directory or all files which have a common `root` name. Other *file name expansion* mechanisms use the metacharacter '~' and allow files in other users' directories to be named easily (2.6, 5.2).

flag Many UNIX System commands accept arguments which are not the names of files or other users but are used to modify the action of the commands. These are referred to as *flag* options, and by convention consist of one or more letters preceded by the character '–' (2.2). Thus the `ls` (list files) command has an option `-s` to list the sizes of files. This is specified:

```
ls –s
```

foreach
The *foreach* command is used in shell scripts and at the terminal to specify repetition of a sequence of commands while the value of a certain shell variable ranges through a specified list (4.6, 5.1).

foreground
When commands are executing in the normal way such that the shell is waiting for them to finish before prompting for another command, they are said to be *foreground jobs* or *running in the foreground*. This is as opposed to *background*. *Foreground* jobs can be stopped by signals from the terminal caused by typing different control characters at the keyboard (2.8, 3.6).

goto The shell has a command *goto* used in shell scripts to transfer control to a given label (4.7).

grep The `grep` command searches through a list of argument files for a specified string. Thus:

```
grep bill /etc/passwd
```

will print each line in the file `/etc/passwd` which contains the string 'bill'. Actually, `grep` scans for *regular expressions* in the sense of the editors *ed*(1) and *ex*(1). `grep` stands for 'globally find *regular expression* and print' (3.4).

history The *history* mechanism of the shell allows previous commands to be repeated, possibly after modification to correct typing mistakes or to change the meaning of the command. The shell has a *history list*, where these commands are kept, and a *history* variable, which controls how large this list is (3.3).

home directory
Each user has a *home directory*, which is given in your entry in the `password` file, `/etc/passwd`. This is the directory that you are placed in when you first log in. The `cd` or `chdir` command with no arguments takes you back to this directory, whose name is recorded in the shell variable *home*. You can also access the *home directories* of other users in forming file names using a *file name expansion* notation and the character tilde (˜) (2.6).

if A conditional command within the shell, the *if* command is used in shell command scripts to make decisions about what course of action to take next (4.6).

ignoreeof
Normally, your shell will exit, displaying 'logout' if you type a CTRL d at the % prompt. This is the way you usually log off the system. You can *set* the *ignoreeof* variable if you wish in your `.login` file and then use the command *logout* to log out. This is useful if you occasionally type too many CTRL d characters, accidentally logging yourself off (3.2).

input Many commands on the UNIX System take information from the terminal or from files which they then act on. This information is called *input*. Commands normally read for *input* from their *standard input* which is, by default, the terminal. This *standard input* can be redirected from a file using a shell metanotation with the character <. Many commands will also read from a file specified as argument. Commands placed in *pipelines* will read from the output of the previous command in the *pipeline*. The leftmost command in a *pipeline* reads from the terminal if you neither redirect its *input* nor give it a filename to use as *standard input*. Special mechanisms exist for supplying input to commands in shell scripts (2.5, 4.8).

interrupt
An *interrupt* is a signal to a program that is generated by typing ^C. (On older versions of the UNIX System, the $\boxed{\text{RUBOUT}}$ or $\boxed{\text{DELETE}}$ keys were used for this purpose.) It causes most programs to stop execution. Certain programs, such as the shell and the editors, handle an *interrupt* in special ways, usually by stopping what they are doing and prompting for another command. While the shell is executing another command and waiting for it to finish, the shell does not listen to *interrupts*. The shell often wakes up when you hit *interrupt* because many commands die when they receive an *interrupt* (2.8, 4.9).

job One or more commands typed on the same input line separated by '|' or ';' characters are run together and are called a *job*. Simple commands run by themselves without any '|' or ';' characters are the simplest *jobs*. *Jobs* are classified as *foreground*, *background*, or *suspended* (3.6).

job control
The builtin functions that control the execution of jobs are called *job control* commands. These are *bg*, *fg*, *stop*, and `kill` (3.6).

job number
When each job is started, it is assigned a small number called a *job number* which is printed next to the job in the output of the *jobs* command. This number, preceded by a '%' character, can be used as an argument to *job control* commands to indicate a specific job (3.6).

jobs The *jobs* command displays a table showing jobs that are either running in the *background* or are *suspended* (3.6).

kill A command which sends a signal to a job causing it to terminate (3.6).

.login The file `.login` in your *home* directory is read by the shell each time you log in to the UNIX System, and the commands found there are executed. There are a number of commands that are usefully placed in this file, especially *set* commands to the shell itself (3.1).

login shell
The shell that is started on your terminal when you log in is called your *login shell*. It is different from other shells which you may run (e.g. on shell scripts) in that it reads the `.login` file before reading commands from the terminal, and it reads the `.logout` file after you log out (3.1).

logout The *logout* command causes a login shell to exit. Normally, a login shell will exit when you type CTRL d generating an *end-of-file,* but if you have set *ignoreeof* in your `.login` file, then this will not work and you must use *logout* to log off the UNIX System (3.8).

.logout When you log off of the UNIX System, the shell will execute commands from the file `.logout` in your *home* directory after it displays 'logout.'

lpr The command *lpr* is the line printer daemon. The standard input of *lpr* is spooled and printed on the UNIX System line printer. You can also give *lpr* a list of filenames as arguments to be printed. It is most common to use *lpr* as the last component of a *pipeline* (3.3).

ls The `ls` (list files) command is one of the most commonly used UNIX System commands. With no argument, it prints the names of the files in the current directory. `ls` has a number of useful *flag* arguments and it can also be given the names of directories as arguments, in which case it lists the names of the files in those directories (2.2).

mail The `mail` program is used to send messages to and receive messages from other UNIX System users (2.1, 3.1), whether they are logged on or not.

make The `make` command is used to maintain one or more related files and to organize functions to be performed on these files. In many ways `make` is easier to use and more helpful than shell command scripts (4.2).

makefile
The file containing commands for `make` is called the `makefile` or `Makefile` (4.2).

manual
The *manual* often referred to is the *INTERACTIVE UNIX System User's/System Administrator's Reference Manual* or the *INTERACTIVE SDS Guide and Programmer's Reference Manual*. They contain numbered sections with a description of each UNIX System program. There are also supplementary documents (tutorials and reference guides) for individual programs which require more detailed explanation. On-line versions of the *INTERACTIVE UNIX System User's/System Administrator's Reference Manual* and the *INTERACTIVE SDS Guide and Programmer's Reference Manual* are accessible through the `man` command. Its documentation can be obtained online via:

man *1 entry*

metacharacter
Many characters which are neither letters nor digits have special meaning either to the shell or to the UNIX System. These characters are called *metacharacters*. If it is necessary to use these characters in arguments to commands and prevent them from having their special meanings, then they must be *quoted*. An example of a *metacharacter* is the character > which is used to indicate placement of output into a file. For the purposes of the *history* mechanism, most unquoted *metacharacters* form separate words (2.4). This glossary lists the *metacharacters* in groups by their function.

mkdir The `mkdir` command is used to create a new directory.

modifier
Substitutions with the *history* mechanism, keyed by the character ! or of variables using the metacharacter $, are often subjected to modifications, indicated by placing the character : after the substitution and following this with the *modifier* itself. The *command substitution* mechanism can also be used to perform modification in a similar way, but this notation is less clear (4.6).

more The program `more` writes a file on your terminal, allowing you to control how much text is displayed at a time. `more` can move through the file screenful by screenful, line by line, search forward for a string, or start again at the beginning of the file. It is generally the easiest way of viewing a file (2.8).

noclobber
The shell has a variable *noclobber* which may be set in the file `.login` to prevent accidental destruction of files by the > output redirection metasyntax of the shell (3.2, 3.5).

noglob The shell variable *noglob* is set to suppress the *file name expansion* of arguments containing the metacharacters ~, *, ?, [, and] (4.6).

notify The *notify* command tells the shell to report on the termination of a specific *background job* at the exact time it occurs, as opposed to waiting until just before the next prompt to report the termination. The *notify* variable, if set, causes the shell to always report the termination of *background* jobs exactly when they occur (3.6).

onintr The *onintr* command is built into the shell and is used to control the action of a shell command script when an *interrupt* signal is received (4.9).

output Many commands in the UNIX System result in some lines of text which are called their *output*. This *output* is usually placed on what is known as the *standard output* which is normally connected to the user's terminal. The shell has a syntax using the metacharacter > for redirecting the *standard output* of a command to a file (2.3). Using the *pipe* mechanism and the metacharacter |, it is also possible for the *standard output* of one command to become the *standard input* of another command (2.5). Certain commands, such as the line printer daemon *p*, do not place their results on the *standard output*, but rather in more useful places such as on the line printer (3.3). Similarly the `write` command places its output on another user's terminal rather than its *standard output* (3.3). Commands also have a *diagnostic output* where they write their error messages. Normally these go to the terminal even if the *standard output* has been sent to a file or another command, but it is possible to direct error diagnostics along with *standard output* using a special metanotation (3.5).

path The shell has a variable, *path*, which gives the names of the directories in which it searches for the commands which it is given. It always checks first to see if the command it is given is built into the shell. If it is, then it need not search for the command, as it can do it internally. If the command is not builtin, then the shell searches for a file with the name given in each of the directories in the *path* variable, left to right. Since the normal definition of the *path* variable is:

```
path     (. /usr/ucb /bin /usr/bin)
```

the shell normally looks in the current directory, and then in the standard system directories `/usr/ucb`, `/bin`, and `/usr/bin` for the named command (3.2). If the command cannot be found, the shell will print an error diagnostic. Scripts of shell commands will be executed using another shell to interpret them if they have 'execute' permission set. This is normally true because a command of the form:

```
chmod 755 script
```

was executed to turn this execute permission on (4.3). If you add new commands to a directory in the *path*, you should issue the command *rehash* (3.2).

path name
A list of names, separated by / characters, forms a *path name*. Each *component*, between successive / characters, names a directory in which the next *component* file resides. *Path names* which begin with the character /

are interpreted relative to the `root` directory in the file system. Other *path names* are interpreted relative to the current directory as reported by `pwd`. The last component of a *path name* may name a directory, but usually names a file.

pipeline
A group of commands which are connected together, the *standard output* of each connected to the *standard input* of the next, is called a *pipeline*. The *pipe* mechanism used to connect these commands is indicated by the shell metacharacter '|' (2.5, 3.3).

popd The *popd* command changes the shell's *working directory* to the directory you most recently left using the *pushd* command. It returns to the directory without having to type its name, forgetting the name of the current *working directory* before doing so (3.7).

port The part of a computer system to which each terminal is connected is called a *port*. Usually the system has a fixed number of *ports*, some of which are connected to telephone lines for dial-up access, and some of which are permanently wired directly to specific terminals.

pr The `pr` command is used to prepare listings of the contents of files with headers, giving the name of the file and the date and time at which the file was last modified (3.3).

process
An instance of a running program is called a *process* (3.6). The UNIX System assigns each *process* a unique number when it is started; it is called the *process number*. *Process numbers* can be used to stop individual *processes* using the *kill* or *stop* commands when the *processes* are part of a detached *background* job.

program
Usually synonymous with *command*; a binary file or shell command script which performs a useful function is often called a *program*.

prompt Many programs will print a *prompt* on the terminal when they expect input. Thus the editor *ex*(1) will print a : when it expects input. The shell *prompts* for input with % and occasionally with ? when reading commands from the terminal (2.1). The shell has a variable *prompt* which may be set to a different value to change the shell's main *prompt*. This is mostly used when debugging the shell (3.8).

pushd The *pushd* command, which means 'push directory,' changes the shell's *working directory* and also remembers the current *working directory* before the change is made, allowing you to return to the same directory via the *popd* command later without retyping its name (3.7).

ps The `ps` command is used to show the processes you are currently running. Each process is shown with its unique process number, an indication of the terminal name it is attached to, an indication of the state of the process (whether it is running, stopped, awaiting some event (sleeping), and whether it is swapped out), and the amount of CPU time it has used so far. The command is identified by printing some of the words used when it was invoked (3.6). Shells, such as the `csh` you use to run the `ps` command, are not normally shown in the output.

pwd The `pwd` command prints the full *path name* of the current *working directory*. The *dirs* builtin command is usually a better and faster choice.

quit The *quit* signal, generated by a CTRL e, is used to terminate programs which are behaving unreasonably. It normally produces a core image file (2.8).

quotation
The process by which metacharacters are prevented their special meaning, usually by using the character ′ in pairs or by using the character \, is referred to as *quotation* (2.7).

redirection
The routing of input to or output from a file is known as *redirection* of input or output (2.3).

rehash The *rehash* command tells the shell to rebuild its internal table of which commands are found in which directories in your *path*. This is necessary when a new program is installed in one of these directories (3.8).

relative path name
A *path name* which does not begin with a / is called a *relative path name* since it is interpreted *relative* to the current *working directory*. The first *component* of such a *path name* refers to some file or directory in the *working directory*, and subsequent *components* between / characters refer to directories below the *working directory*. *Path names* that are not *relative* are called *absolute path names* (2.6).

repeat The *repeat* command iterates another command a specified number of times.

root The directory that is at the top of the entire directory structure is called the `root` directory since it is the 'root' of the entire tree structure of directories. The name used in *path names* to indicate the `root` is '/'. *Path names* starting with / are said to be *absolute* since they start at the `root` directory. `root` is also used as the part of a *path name* that is left after removing the *extension*. See *file name* for a further explanation (2.6).

RUBOUT
The **RUBOUT** or **DELETE** key is often used to erase the previously typed character; some users prefer the **BACKSPACE** for this purpose. On older versions of the UNIX system this key served as the INTR character.

scratch file
Files that have names beginning with a # are referred to as *scratch files*, since they are automatically removed by the system after a couple of days of non-use, or more frequently if disk space becomes tight (2.3).

script Sequences of shell commands placed in a file are called shell command *scripts*. It is often possible to perform simple tasks using these *scripts*, without writing a program in a language such as C, by using the shell to selectively run other programs (4.3, 4.10).

set The builtin *set* command is used to assign new values to shell variables and to show the values of the current variables. Many shell variables have special meaning to the shell itself. Thus by using the *set* command the behavior of the shell can be affected (3.1).

setenv Variables in the environment *environ*(5) can be changed by using the *setenv* builtin command (3.8). The *setenv* command can also be used to print the values of the variables in the environment.

shell A *shell* is a command language interpreter. It is possible to write and run your own *shell*, as *shell*s are no different than any other programs as far as the system is concerned. This manual deals with the details of one particular *shell*, called `csh`.

shell script
See *script* (4.3, 4.10).

signal A *signal* in the UNIX System is a short message that is sent to a running program which causes something to happen to that process. *Signals* are sent either by typing special *control* characters on the keyboard or by using the `kill` or *stop* commands (2.8, 3.6).

sort The `sort` program sorts a sequence of lines in ways that can be controlled by argument *flags* (2.5).

source The *source* command causes the shell to read commands from a specified file. It is most useful for reading files, such as `.cshrc`, after changing them (3.8).

special character
See *metacharacters*.

standard
We refer often to the *standard input* and *standard output* of commands. See *input* and *output* (2.3, 4.8).

status A command normally returns a *status* when it finishes. By convention a *status* of zero indicates that the command succeeded. Commands may return nonzero *status* to indicate that some abnormal event has occurred. The shell variable *status* is set to the *status* returned by the last command. It is most useful in shell command scripts (4.6).

stop The *stop* command causes a *background* job to become *suspended* (3.6).

string A sequential group of characters taken together is called a *string*. *Strings* can contain any printable characters (3.2).

stty The `stty` program changes certain parameters inside the UNIX System which determine how your terminal is handled. See *stty*(1) for a complete description (3.6).

substitution
The shell implements a number of *substitutions* where sequences indicated by metacharacters are replaced by other sequences. Notable examples of this are history *substitution* keyed by the metacharacter '!' and variable *substitution* indicated by '$'. We also refer to *substitutions* as *expansions* (4.4).

suspended
A job becomes *suspended* after a STOP signal is sent to it, either by typing a $\boxed{\text{CTRL}}$ $\boxed{\text{z}}$ at the terminal (for *foreground* jobs) or by using the *stop* command (for *background* jobs). When *suspended*, a job temporarily stops running until it is restarted by either the *fg* or *bg* command (3.6).

switch The *switch* command of the shell allows the shell to select one of a number of sequences of commands based on an argument string. It is similar to the *switch* statement in the language C (4.7).

termination
When a command which is being executed finishes, we say it undergoes *termination* or *terminates*. Commands normally terminate when they read an *end-of-file* from their *standard input*. It is also possible to terminate commands by sending them an *interrupt* or *quit* signal (2.8). The `kill` program terminates specified jobs (3.6).

then The *then* command is part of the shell's 'if-then-else-endif' control construct used in command scripts (4.6).

time The `time` command can be used to measure the amount of CPU and real time consumed by a specified command. (3.1, 3.8).

tset The `tset` program is used to set standard erase and kill characters and to tell the system what kind of terminal you are using. It is often invoked in a `.login` file (3.1).

tty The word *tty* is a historical abbreviation for 'teletype' which is frequently used in the UNIX System to indicate the *port* to which a given terminal is connected. The `tty` command will print the name of the *tty* or *port* to which your terminal is presently connected.

unalias
The *unalias* command removes aliases (3.8).

UNIX UNIX is an operating system on which `csh` runs. The UNIX System provides facilities which allow `csh` to invoke other programs such as editors and text formatters which you may wish to use.

unset The *unset* command removes the definitions of shell variables (3.2, 3.8).

variable expansion
See *variables* and *expansion* (3.2, 4.4).

variables
Variables in `csh` hold one or more strings as value. The most common use of *variables* is in controlling the behavior of the shell. See *path*, *noclobber*, and *ignoreeof* for examples. *Variables* such as *argv* are also used in writing shell programs (shell command scripts) (3.2).

verbose The *verbose* shell variable can be set to cause commands to be echoed after they are history expanded. This is often useful in debugging shell scripts. The *verbose* variable is set by the shell's `-v` command line option (4.10).

wc The `wc` program calculates the number of characters, words, and lines in the files whose names are given as arguments (3.6).

while The *while* builtin control construct is used in shell command scripts (4.7).

word A sequence of characters which forms an argument to a command is called a *word*. Many characters which are neither letters, digits, '–', '.', nor '/' form *words* all by themselves even if they are not surrounded by blanks. Any sequence of characters may be made into a *word* by surrounding it with '" characters, except for the characters '" and '!' which require special treatment (2.1). This process of placing special characters in *words* without their special meaning is called *quoting*.

working directory
At any given time you are in one particular directory, called your *working directory*. This directory's name is printed by the `pwd` command and the files listed by `ls` are the ones in this directory. You can change *working directories* using `chdir`.

write The `write` command is an obsolete way of communicating with other users who are logged in to the UNIX System (you have to take turns typing). If you are both using display terminals, use *talk*(1), which is much more pleasant.

POSIX Compliance Document

This document is required by the Institute of Electrical and Electronic Engineers (IEEE) for an implementation claiming conformance to *IEEE Std. 1003.1-1988* (section 2.2.1.2). It has the same structure as the standard, with the information presented in the appropriately numbered sections.

CONFORMANCE STATEMENT

The INTERACTIVE UNIX* Operating System Version 3.0 conforms with the following standards:

The Institute of Electrical and Electronic Engineers Standard for Portable Operating System Interface for Computing Environments, *IEEE Std. 1003.1-1988*.

Announcing the Standard for POSIX: PORTABLE OPERATING SYSTEM INTERFACE FOR COMPUTER ENVIRONMENTS. Federal Information Processing Standards Publication 151-1.

The contents of the `limits.h` header is described under section 2.9, and the contents of the `unistd.h` header under section 2.10.

2.2.1.3 CONFORMING IMPLEMENTATION OPTIONS

The INTERACTIVE UNIX Operating System uses the following options:

Option	*Value*	*Comment*
{NGROUPS_MAX}	16	Multiple groups option
{_POSIX_JOB_CONTROL}	1	Job control option
{_POSIX_CHOWN_RESTRICTED}	1	Administrative/security option

2.5 ERROR NUMBERS

The following symbolic names identify the possible error numbers and their INTERACTIVE UNIX System-defined values:

Symbolic Name	*Value*	*Comment*
[E2BIG]	7	Arg list too long
[EACCES]	13	Permission denied
[EAGAIN]	11	No more processes
[EBADF]	9	Bad file number
[EBUSY]	16	Mount device busy
[ECHILD]	10	No children
[EDEADLK]	45	Deadlock condition
[EDOM]	33	Math arg out of domain of func
[EEXIST]	17	File exists
[EFAULT]	14	Bad address
[EFBIG]	27	File too large
[EINTR]	4	Interrupted system call
[EINVAL]	22	Invalid argument
[EIO]	5	I/O error
[EISDIR]	21	Is a directory
[EMFILE]	24	Too many open files
[EMLINK]	31	Too many links
[ENAMETOOLONG]	78	Path or path component exceeds limit
[ENFILE]	23	File table overflow
[ENODEV]	19	No such device
[ENOENT]	2	No such file or directory
[ENOEXEC]	8	Exec format error
[ENOLCK]	46	No record locks available.
[ENOMEM]	12	Not enough core
[ENOSPC]	28	No space left on device
[ENOSYS]	89	Function not implemented
[ENOTDIR]	20	Not a directory
[ENOTEMPTY]	93	Directory not empty
[ENOTTY]	25	Not a typewriter
[ENXIO]	6	No such device or address
[EPERM]	1	Not superuser
[EPIPE]	32	Broken pipe
[ERANGE]	34	Math result not representable
[EROFS]	30	Read only file system
[ESPIPE]	29	Illegal seek
[ESRCH]	3	No such process
[EXDEV]	18	Cross-device link

2.6 PRIMITIVE SYSTEM DATA TYPES

The following data types are used by the various system functions. POSIX requires the following:

Data Type	*Value*	*Comment*
dev_t	short	<old device number> type
gid_t	unsigned short	
ino_t	ushort	<inode> type
mode_t	unsigned short	
nlink_t	short	
off_t	long	<offset> type
pid_t	short	
uid_t	unsigned short	

The data types that are defined by the INTERACTIVE UNIX Operating System and not by the standard are:

Data Type	*Value*	*Comment*
cnt_t	short	<count> type
time_t	long	<time> type
label_t[6]	int	
paddr_t	unsigned long	<physical address> type
key_t	int	IPC key type
use_t	unsigned char	use count for swap
sysid_t	short	
index_t	short	
lock_t	short	lock work for busy wait
size_t	unsigned int	len param for string funcs
sel_t	ushort	selector type

2.8.2.2 Common Usage C Language Dependent System Support

The INTERACTIVE UNIX Operating System is delivered with a common C compiler, rather than with an ANSI C compiler. The header files contain support for an ANSI C compiler environment when one becomes available.

Compilation of processes for the POSIX environment is enabled with the new `-Xp` option to the C compiler. See *cc*(1) for details.

If _POSIX_SOURCE is not defined before headers are included, many POSIX symbols will not be available. Therefore, users should be careful always to define _POSIX_SOURCE at the beginning of each program module.

2.9 NUMERICAL LIMITS

The following values are assigned to the symbolic names defined in the header file `limits.h`.

2.9.1 C Language Limits

The following limits are defined in `limits.h`:

Symbolic Limit	*Value*
CHAR_BIT	8
SCHAR_MIN	(-128)
SCHAR_MAX	127
UCHAR_MAX	255
CHAR_MIN	SCHAR_MIN
CHAR_MAX	SCHAR_MAX
MB_LEN_MAX	1
SHRT_MIN	(-32768)
SHRT_MAX	32767
USHRT_MAX	65535
INT_MIN	(-2147483647-1)
INT_MAX	2147483647
UINT_MAX	4294967295
LONG_MIN	(-2147483647L-1)
LONG_MAX	2147483647L
ULONG_MAX	4294967295L

The values of INT_MIN and LONG_MIN are constant expressions, because the constant 2147483648 would have the incorrect type "unsigned long int."

2.9.2 Minimum Values

The following minimum values are defined in `limits.h`:

Symbolic Limit	*Value*
_POSIX_ARG_MAX	4096
_POSIX_CHILD_MAX	6
_POSIX_LINK_MAX	8
_POSIX_MAX_CANON	255

Symbolic Limit	*Value*
_POSIX_MAX_INPUT	255
_POSIX_NAME_MAX	14
_POSIX_NGROUPS_MAX	0
_POSIX_OPEN_MAX	16
_POSIX_PATH_MAX	255
_POSIX_PIPE_BUF	512

2.9.3 Run-Time Increasable Values

The following run-time increasable value is defined in `limits.h`:

Run-Time Increasable	*Value*
NGROUPS_MAX	16

The number of supplementary groups is a tunable parameter. The user can change the number of groups supported by a particular kernel at configuration time. The default is 16. The upper bound is dependent only on system resources. Use the `sysconf()` function to determine the current setting.

2.9.4 Run-Time Invariant Values

The following run-time invariant value is defined in `limits.h`:

Run-Time Invariant	*Value*
ARG_MAX	5120

Neither CHILD_MAX or OPEN_MAX are defined in `limits.h`.

The maximum number of simultaneous processes per real user ID is a tunable paramenter that can be changed at kernel configuration time. The minimum value required is 6; the default value is 25. Use the `sysconf()` function to determine the current value of CHILD_MAX.

The maximum number of files that any process can have open at a given time is a tunable parameter that can be changed at kernel configuration time. The minimum value required is 16; the default value is 20. Use the `sysconf()` function to determine the current value of OPEN_MAX.

2.9.5 Pathname Variable Values

The following pathname variable values are defined in `limits.h`:

Pathname Variable	*Value*
LINK_MAX	1000
MAX_CANON	255
MAX_INPUT	255
NAME_MAX	14
PATH_MAX	256
PIPE_BUF	10240

In addition, the following symbolic name is defined in `limits.h`:

`NULL` 0

2.10 SYMBOLIC CONSTANTS

The following values are assigned to the symbolic names defined in the header file `unistd.h`.

2.10.1 Symbolic Constants for the access() Function

The following symbolic constants for the `access()` function are defined in `unistd.h`:

Symbolic Constant	*Value*	*Comment*
R_OK	4	Test for Read permission
W_OK	2	Test for Write permission
X_OK	1	Test for eXecute permission
F_OK	0	Test for existence of File

2.10.2 Symbolic Constants for the lseek() Function

The following symbolic constants for the `lseek()` function are defined in `unistd.h`:

Symbolic Constant	*Value*	*Comment*
SEEK_SET	0	Set file pointer to "offset"
SEEK_CUR	1	Set file pointer to current plus "offset"
SEEK_END	2	Set file pointer to EOF plus "offset"

2.10.3 Compile-Time Symbolic Constants For Portability Specifications

The following compile-time symbolic constants for portability specifications are defined in `unistd.h`:

Compile-Time Symbolic	*Value*	*Comment*
_POSIX_JOB_CONTROL	1	have job control
_POSIX_SAVED_IDS	1	have saved ids
_POSIX_VERSION	198808L	

2.10.4 Execution-Time Symbolic Constants For Portability Specifications

The following execution-time symbolic constants for portability specifications are defined in `unistd.h`:

Execution-Time Symbolic	*Value*
_POSIX_CHOWN_RESTRICTED	1
_POSIX_NO_TRUNC	1
_POSIX_VDISABLE	255

In addition, the following symbolic constants are defined in `unistd.h` for `pathconf()`:

Symbolic Constant	*Value*
_PC_LINK_MAX	1
_PC_MAX_CANON	2
_PC_MAX_INPUT	3
_PC_NAME_MAX	4
_PC_PATH_MAX	5
_PC_PIPE_BUF	6
_PC_CHOWN_RESTRICTED	7
_PC_NO_TRUNC	8
_PC_VDISABLE	9

The following symbolic constants are also defined in `unistd.h` for `sysconf()`:

Symbolic Constant	*Value*
_SC_ARG_MAX	1
_SC_CHILD_MAX	2
_SC_CLK_TCK	3
_SC_NGROUPS_MAX	4
_SC_OPEN_MAX	5
_SC_JOB_CONTROL	6
_SC_SAVED_IDS	7
_SC_VERSION	8
STDIN_FILENO	0
STDOUT_FILENO	1
STDERR_FILENO	2

3.1.2 Execute a File

If the environment variable PATH is not set, the path used for `execlp()` and `execvp()` for the file argument is `:/bin:/usr/bin`, meaning that the current directory is searched, then `/bin`, then `/usr/bin`.

3.3.1.1 Signal Names

The following constants are defined by the INTERACTIVE UNIX Operating System in `signal.h` for this POSIX implementation:

Signal Name	*Value*	*Comment*
SIGHUP	1	hangup
SIGINT	2	interrupt
SIGQUIT	3	quit
SIGILL	4	illegal instruction
SIGTRAP	5	trace trap
SIGIOT	6	IOT instruction
SIGABRT	6	used by abort, replace SIGIOT in the future
SIGEMT	7	EMT instruction
SIGFPE	8	floating point exception
SIGKILL	9	kill (cannot be caught or ignored)
SIGBUS	10	bus error
SIGSEGV	11	segmentation violation
SIGSYS	12	bad argument to system call
SIGPIPE	13	write on a pipe with no one to read it
SIGALRM	14	alarm clock

Signal Name	*Value*	*Comment*
SIGTERM	15	software termination signal from kill
SIGUSR1	16	user defined signal 1
SIGUSR2	17	user defined signal 2
SIGCHLD	18	death of a child
SIGCLD	18	death of a child (older spelling of SIGCHLD)
SIGPWR	19	power-fail restart
SIGWINCH	20	window change
SIGPOLL	22	pollable event occurred
		Job control signals (other than SIGCHLD, defined above)
SIGCONT	23	continue if stopped
SIGSTOP	24	stop signal (cannot be caught or ignored)
SIGTSTP	25	interactive stop signal
SIGTTIN	26	background read attempted
SIGTTOU	27	background write attempted

3.3.1.2 Signal Generation and Delivery

If a subsequent occurrence of a pending signal is generated, it will not be delivered more than once, that is, signals are not counted or queued.

3.3.2 Send a Signal to a Process

In compliance with FIPS 151-1, the saved `set_user_id` of the receiving function is checked in place of the effective user ID to determine if the sending process has the permission to send a signal to a process.

3.3.4.2 Examine and Change Signal Action

The following symbolic constant is defined in `signal.h`:

Constant	*Value*	*Comment*
SA_NOCLDSTOP	1	Do not generate SIGCHLD when children stop

3.3.5.2 Examine and Change Blocked Signals

The following symbolic constants are defined in `signal.h`:

Constant	*Value*	*Comment*
SIG_BLOCK	0	Add to the set of blocked signals
SIG_UNBLOCK	1	Subtract from the set of blocked signals
SIG_SETMASK	2	Set the blocked signal set

4.2.2 Set User and Group IDs

In compliance with FIPS 151-1, the saved `set_user_id` and saved `set_group_id` are set by these functions.

4.3.3 Set Process Group ID for Job Control

{POSIX_JOB_CONTROL} is defined, therefore the `setpgid()` function allows a process to join or create a process group.

The [ENOSYS] error will not be generated, as `setpgid` is fully supported.

4.4.1 System Name

The standard does not specify any error conditions that are required to be detected for the `uname()` function. The INTERACTIVE UNIX Operating System will not generate any additional errors.

4.5.1 Get System Time

The standard does not specify any error conditions that are required to be detected for the `time()` function. The INTERACTIVE UNIX Operating System may return [EFAULT] if the pointer specified by the argument `tloc` refers to memory that is not writable by the user process.

4.5.2 Process Times

The standard does not specify any error conditions that are required to be detected for the `times()` function. The INTERACTIVE UNIX Operating System may return [EFAULT] if the pointer specified by the argument `buffer` refers to memory that is not writable by the user process.

4.6.1 Environment Access

The standard does not specify any error conditions that are required to be detected for the `getenv()` function. The INTERACTIVE UNIX Operating System will not generate any additional errors.

4.7.1 Generate Terminal Pathname

The standard does not specify any error conditions that are required to be detected for the `ctermid()` function. The INTERACTIVE UNIX Operating System will not generate any additional errors.

4.7.2 Determine Terminal Device Name

The standard does not specify any error conditions that are required to be detected for the `ttyname()` or `isatty()` functions. The INTERACTIVE UNIX Operating System will not generate any additional errors.

5.3.1 Open a File

In compliance with FIPS 151-1, when a file is created, its group is set to the group of the directory in which the file is created.

5.3.3.2 Set File Creation Mask

Bits other than file protection bits are silently ignored.

5.3.4 Link to a File

The `link` function is allowed for files that are not accessible. Links are not supported across file systems.

5.4.1 Make a Directory

When mode bits other than the file permission bits are set, they are silently ignored. They do not affect the created directory.

5.4.2 Make a FIFO Special File

When mode bits other than the file permission bits are set, they are silently ignored. They do not affect the created FIFO.

5.5.1 Remove Directory Entries

The INTERACTIVE UNIX Operating System supports using the `unlink()` function for directories if the user has appropriate privileges.

5.6.3 File Accessibility

If there are no execution bits set for a file, X_OK will not be returned.

5.6.4 Change File Modes

There are no additional restrictions for the INTERACTIVE UNIX Operating System on setting S_ISUID and S_ISGID mode bits.

5.6.5 Change Ownership and Group of a File

If the path argument refers to a regular file, the set-user-ID (S_ISUID) and set-group-ID (S_ISGID) bits of the file mode shall be cleared upon successful return from `chown()`, without regard to appropriate privileges.

6.4.1 Read From a File

In compliance with FIPS 151-1, if `read()` is interrupted by a signal after it has successfully read some data, it returns the number of bytes the system has read.

6.4.2 Write to a File

In compliance with FIPS 151-1, if `write()` is interrupted by a signal after it has successfully written some data, it returns the number of bytes the system has written.

6.5.2 File Control

Additional bits used for file modes other than permissions with the F_SETFL command are silently ignored.

7.1.1.3 The Controlling Terminal

If a session leader has no controlling terminal and opens a terminal device file that is not already associated with a session without using the O_NOCTTY option, the terminal becomes the controlling terminal of the session leader.

7.1.1.9 Special Characters

Start and stop characters can be changed. No multi-byte functions are supported with IEXTEND.

7.1.2.7.4 Baud Rate Function Description

The standard does not specify any error conditions that are required to be detected for the baud rate functions. The INTERACTIVE UNIX Operating System will not generate any additional errors.

9.2.1 Group Database Access

The standard does not specify any error conditions that are required to be detected for the `getgrid()` or `getgrnam()` functions. The INTERACTIVE UNIX Operating System will not generate any additional errors.

9.2.2 User Database Access

The standard does not specify any error conditions that are required to be detected for the `getpwnam()` or `getpwnam()` functions. The INTERACTIVE UNIX Operating System will not generate any additional errors.

10.1 ARCHIVE/INTERCHANGE FILE FORMAT

The format-creating utility is used to translate from the file system to the formats defined in this section in a implementation-defined way, and the format-reading utility is used to translate from the formats defined in this section to a file system.

The user-level program *pax*(1P) is used to provide the archive and interchange file format reading and writing. It can read and write both `tar` and `cpio` formats. The full syntax for this command is documented in *pax*(1P), which was written by Mark H. Colburn and sponsored by the USENIX Association for public distribution.

It should be noted that the traditional SVR3 programs, `tar` and `cpio`, have been retained in this distribution for compatibility with older releases. In particular, the `tar` program does not read or write the new `tar` archive format. The new `pax` program should be used to read and write archives using the new file interchange formats.

Documentation Roadmap

CONTENTS

Documentation Roadmap

OVERVIEW

This "Documentation Roadmap" describes the complete documentation set available for the INTERACTIVE™ Product Family from SunSoft. Two types of documentation are delivered with most products: user-level documentation and technical reference guides.

The INTERACTIVE UNIX® Operating System is delivered with five guides. The first is for users who are new to UNIX Systems and is called *INTERACTIVE UNIX System Guide for New Users*. This provides a primer that contains a basic overview of the structure and commands used in the INTERACTIVE UNIX System along with numerous examples. It also contains "System Administration for New Users of the INTERACTIVE UNIX Operating System," which describes the basic system administration procedures needed to manage the INTERACTIVE UNIX Operating System.

The second guide is called the *INTERACTIVE UNIX Operating System Guide*. This guide contains the information needed to install the INTERACTIVE UNIX Operating System and a document that describes advanced system administration procedures.

The third guide is called the *International Supplement Guide*. This guide contains the information needed to prepare and install the International Supplement and provides guidelines for developing internationalized applications.

The fourth guide is called the *INTERACTIVE UNIX System User's/System Administrator's Reference Manual*. This guide contains the manual entries for the commands that constitute the basic software installed with the INTERACTIVE UNIX System.

The fifth guide is called the *INTERACTIVE UNIX System User's Guide*. This guide contains general information about the UNIX Operating System as well as tutorials on certain UNIX System commands.

The documentation delivered with each software package is described in more detail in subsequent sections. Most of these guides may be ordered separately from SunSoft or a SunSoft distributor.

INTERACTIVE PRODUCT FAMILY DOCUMENTATION

This section describes both the standard and optional documentation available for the INTERACTIVE Product Family. Books are listed by product extension. Please note that not all extensions are available as separate products.

INTERACTIVE UNIX System V/386 Release 3.2

Standard Documentation

INTERACTIVE UNIX System V/386 Release 3.2
Guide for New Users

Describes the basic structure, commands, and use of the INTERACTIVE UNIX Operating System and the basic `sysadm` and other system administration procedures needed to manage the system.

INTERACTIVE UNIX System V/386 Release 3.2
Operating System Guide

In separate documents it introduces the reader to the INTERACTIVE Product Family, details the installation of the INTERACTIVE UNIX Operating System, and describes advanced maintenance procedures. It also includes:

- *Sendmail Documents*
  Describe how to install, customize, and use the Berkeley `sendmail` mail routing facility.

- *C Shell Document*
  Provides a basic introduction to the Berkeley C shell, including shell startup and termination, commands, redirection of input and output, metacharacters, pipelining, file names, quoting, variables, aliases, job control, built-in commands, a glossary, and shell control structures and command scripts.

- *POSIX Compliance Document*
  Describes the conformance of the INTERACTIVE UNIX Operating System to the POSIX/FIPS standard as specified by *IEEE Std. 1003.1-1988* and FIPS 151-1.

International Supplement Guide

Provides information about how to prepare and install the International Supplement on an INTERACTIVE UNIX Operating System, summarizes the internationalization features, and provides programming tips for developing internationalized applications. This guide also contains the XPG conformance document.

INTERACTIVE UNIX System V/386 Release 3.2
User's/System Administrator's Reference Manual

Contains the USL UNIX System and INTERACTIVE UNIX System manual entries used by system administrators and users.

INTERACTIVE UNIX System V/386 Release 3.2
User's Guide

Introduces users to UNIX System V. It provides a general description of the UNIX System and a number of tutorials, including `ed`, `sh`, `vi`, `csh`, and `mail`.

This document is a reprint of a USL manual and is also published by Prentice Hall and available in most computer bookstores. The title is the same but with "INTERACTIVE" removed.

Optional Documentation

None

VP/ix Environment

Standard Documentation

VP/ix Environment Guide

Describes how to install, use, and maintain the VP/ix™ Environment. It includes a basic primer for using VP/ix and for using MS-DOS® (DOS) under the VP/ix Environment. Also included are SunSoft's proprietary manual entries that supplement the *INTERACTIVE UNIX System User's/System Administrator's Reference Manual*.

Optional Documentation

None

TEN/PLUS Environment

Standard Documentation

The TEN/PLUS Environment comes with a single binder that consists of two guides. Note that in the solution bundles the VP/ix documentation and the TEN/PLUS documentation share the same binder.

TEN/PLUS User Interface Guide
Describes how to use the TEN/PLUS User Interface to create, edit, and manage text files. It contains a self-paced training guide for beginners, a detailed reference manual, a keyboard reference manual that describes the various keyboards supported by the TEN/PLUS system, a TEN/PLUS profiles guide for those who wish to customize their TEN/PLUS Environment, and manual entries describing user commands and files included with the TEN/PLUS User Interface.

TEN/PLUS Mail System Guide
This guide includes a description of how to install and use the TEN/PLUS Mail System. It contains a tutorial and a reference manual that describe how to use and customize the system for reading and sending electronic mail. The guide also explains how to use the system to exchange electronic mail with users on other computers. A set of manual entries describes the commands included with the Mail System, and an installation guide explains how to install and maintain the mail.

Optional Documentation

None

INTERACTIVE Software Development System

Standard Documentation

INTERACTIVE Software Development System Guide and Programmer's Reference Manual
A two-volume document that describes how to install and use the INTERACTIVE Software Development System (SDS). It also contains all the manual entries (from both USL and SunSoft) that describe the programming features of the INTERACTIVE UNIX Operating System, including commands, system calls, subroutines, libraries, file formats, macro packages, and character set tables.

UNIX System V/386 Release 3.2 Programmer's Guide

A two-volume document that describes the UNIX System programming environment. It provides detailed descriptions or tutorials on a variety of programming tools, including `make`, SCCS, `awk`, generating shared libraries, using `curses/terminfo`, and how the UNIX System interfaces with a programming language. It does not include descriptions of any programming languages.

This document is a reprint of a USL manual and is also published by Prentice Hall and available in most computer bookstores under the same title.

UNIX System V/386 Release 3.2 Integrated Software Development Guide

Provides information needed to write applications software and installable drivers for UNIX System V/386 Release 3.2.

This document is a reprint of a USL manual and is also published by Prentice Hall and available in most computer bookstores under the same title.

UNIX System V/386 Release 3.2 SDS Release Notes

Provides an overview of the SDS software and describes its features. It also includes documentation updates.

This document is a reprint of a USL manual and is also published by Prentice Hall and available in most computer bookstores under the same title.

New C Reference Manual

Contains the ''User's Guide,'' which describes how to use the LPI-C™ compiler, and the ''Language Reference Manual,'' which describes the ANSI C language as implemented by LPI.

CodeWatch Reference Manual

Provides general information about using CodeWatch™, an interactive source level debugger.

CoEdit Reference Manual

Describes how to use the CoEdit™ program editor and contains a list and description of the CoEdit commands.

Optional Documentation

UNIX System V/386 Release 3.2 Network Programmer's Guide

Provides an introduction and overview of the USL Transport Interface, its capabilities, and its applications.

This document is a reprint of a USL manual and is also published by Prentice Hall and available in most computer bookstores under the same title.

UNIX System V/386 Release 3.2 STREAMS Primer

Provides a technical overview of STREAMS, including a summary of the STREAMS mechanism, a description of the applications and benefits of STREAMS, and a discussion of each of the facilities provided by STREAMS.

This document is a reprint of a USL manual and is also published by Prentice Hall and available in most computer bookstores under the same title.

UNIX System V/386 Release 3.2 STREAMS Programmer's Guide

A two-part manual that describes how to use STREAMS facilities. Part 1 provides design and programming information for applications programmers developing user-level STREAMS applications. Part 2 provides systems programmers with the information they need to use STREAMS facilities to write UNIX System kernel modules and device drivers. It provides detailed information on the development methods and design philosophy of all aspects of STREAMS. Also included are appendices, which contain a summary of the kernel-level data structures, STREAMS message types, and specifications of kernel utility routines.

This document is a reprint of a USL manual and is also published by Prentice Hall and available in most computer bookstores under the same title.

INTERACTIVE TCP/IP

Standard Documentation

INTERACTIVE TCP/IP Guide

Describes how to install and use the TCP/IP Ethernet support and includes information on two programming interfaces of the Transmission Control Protocol/Internet Protocol (TCP/IP): USL

Transport Layer Interface (TLI) and Berkeley Software Distribution (BSD). Includes a primer that describes user-level networking commands for accessing and using resources on remote hosts.

INTERACTIVE Network Drivers Overview and Installation Instructions

Describes how to install and configure the network drivers on your system. This document is not available separately.

Optional Documentation

None

INTERACTIVE NFS Extension

Standard Documentation

INTERACTIVE NFS Guide

Contains a user's manual that introduces the commands needed by users new to the NFS® distributed computing file system and an installation and maintenance document that provides step-by-step instructions for installing, initializing, and maintaining the NFS system. Also included are user, programmer, and system administrator manual entries. Supplemental documentation provides an overview of protocol specifications.

Optional Documentation

INTERACTIVE Network Information Service Guide
(formerly distributed as *INTERACTIVE Yellow Pages Guide*)

Contains the information needed to install, initialize, and maintain the INTERACTIVE Network Information Services optional subset.

INTERACTIVE X11 Runtime System

Standard Documentation

INTERACTIVE X11 Runtime System Guide

Provides the information about how to maintain and use the INTERACTIVE X11 Runtime System, and includes the Easy Windows™ Environment and the Open Software Foundation™ (OSF)™ Motif™ Window Manager (MWM).

O'Reilly & Associates X Window System User's Guide, Motif Edition
Provides information about how to use X.

Optional Documentation

None

INTERACTIVE X11 Development System

Standard Documentation

INTERACTIVE X11 Development System Guide
Provides two documents that supplement the four O'Reilly and Associates manuals supplied with the INTERACTIVE X11 Development System. The "Inter-Client Communication Conventions Manual" was developed by the MIT X Consortium. It defines standards for a client that converses with a server using the X Window System™ Version 11 protocol. The "INTERACTIVE TCP/IP Programmer's Supplement" is reprinted from the *INTERACTIVE TCP/IP Guide.* It provides supplemental information on how to program the USL Transport Layer Interface and the Berkeley Software Distribution socket interface.

O'Reilly & Associates Xlib Programming Manual
Provides information about the X library, the C language programming interface of the X Window System. It includes a conceptual introduction, tutorial material, and programming examples.

O'Reilly & Associates Xlib Reference Manual
Contains the manual entries for the X library.

O'Reilly & Associates X Toolkit Intrinsics Programming Manual, Motif Edition
Describes how to use the X Toolkit routines.

O'Reilly & Associates X Toolkit Intrinsics Reference Manual
Contains reference pages for the X Toolkit functions.

Optional Documentation

None

INTERACTIVE Motif Development System

Standard Documentation

INTERACTIVE Motif Development System Installation Instructions
Provides information about how to install the INTERACTIVE Motif Development System, derived from OSF/Motif™, Revision 1.1.1, licensed by SunSoft from the Open Software Foundation, Inc.

O'Reilly & Associates Motif Programming Manual
Provides information about how to write applications using the tools supplied with the Motif Development System.

Prentice Hall OSF/Motif Style Guide
Provides information useful to developers about the OSF/Motif interface.

Optional Documentation

None

INTERACTIVE Looking Glass Professional Desktop Manager

Standard Documentation

INTERACTIVE Looking Glass Professional User's Guide
Provides information about how to maintain and use the INTERACTIVE Looking Glass Professional™ Desktop Manager, derived from the product by Visix Software, Inc.

Optional Documentation

None

INTERACTIVE SMB/ix Extension

Standard Documentation

INTERACTIVE SMB/ix Guide
Describes how to install and use the INTERACTIVE SMB/ix® Extension, derived from SMB/ix as developed by Micro Computer Systems, Inc.

INTERACTIVE SMB/ix UNIX System Client Support Guide
Contains release notes and manual entries for the INTERACTIVE SMB/ix UNIX System Client Support.

Optional Documentation

None

INTERACTIVE Security Extension

Standard Documentation

INTERACTIVE Security Guide
Provides information about how to maintain and use the optional INTERACTIVE Security™ Extension, derived from the SMP™ and/or SMP+™ products licensed by SunSoft from SecureWare, Inc.

Optional Documentation

None

FOR MORE INFORMATION

To order the guides described on the previous pages, contact your sales representative.

In addition to the guides described in this document, there are hundreds of commercial books now available that describe various aspects of the UNIX Operating System and the C programming language. A short list of recommended titles follows. There are many other excellent titles; this is just a representative sample. Contact your local technical bookstore for additional recommendations.

UNIX Operating System:

Beginner:

Grace Todino and John Strang, *Learning the UNIX Operating System*, O'Reilly & Associates, Inc.

Peter Birns, P. Brown, and J. C. C. Muster, *UNIX for People*, Prentice Hall.

James R. Groff and Paul N. Weinberg, *Understanding UNIX: A Conceptual Guide*, Que Corporation.

Intermediate:

S. R. Bourne, *The UNIX System*, Addison-Wesley Publishing Co.

Kaare Christian, *The UNIX Operating System*, John Wiley and Sons.

Advanced:

The Waite Group, *UNIX Papers for UNIX Developers and Power Users*, Howard W. Sams and Company.

Maurice Bach, *The Design of the UNIX System*, Prentice Hall.

C Programming:

Beginner:

Morris I. Bolsky and David G. Korn, *The KornShell Command and Programming Language*, Prentice Hall.

Steve Oualline, *Practical C Programming*, O'Reilly & Associates, Inc.

Thomas Plum, *Learning to Program in C*, Prentice Hall.

Charles A. Stanley, *C Language for Beginners*, Pressure Applications.

Intermediate:

John Strang, *Programming with curses*, O'Reilly & Associates, Inc.

Brian W. Kernighan and Dennis M. Ritchie, *The C Programming Language, Second Edition*, Prentice Hall.

Marc Rochkind, *Advanced UNIX Programming*, Prentice Hall.

Advanced:

Naraine Gehani, *Advanced C: Food for the Educated Palate*, Computer Science Press, distributed by W. H. Freeman.

Brian W. Kernighan and Robert Pike, *UNIX Programming Environment*, Prentice Hall.

X Window System:

Beginner:

OSF/Motif User's Guide, Prentice Hall.

Intermediate:

Edited by Tim O'Reilly & Daniel Gilly, *The X Window System in a Nutshell*, O'Reilly & Associates, Inc.

Paul J. Asente and Ralph R. Swick, *X Window System Toolkit: The Complete Programmer's Guide and Specification*, Prentice Hall.

Robert W. Scheifler and James Gettys, *X Window System: The Complete Reference to Xlib, Protocol, ICCCM, XLFD, Second Edition*, Prentice Hall.

Advanced:

Edited and with an introduction by Adrian Nye, *X Protocol Reference Manual*, O'Reilly & Associates, Inc.

OSF/Motif Programmer's Reference, Prentice Hall.

System Administration:

Beginner:

Eric Foxley, *UNIX for Super-Users*, Addison-Wesley Publishing Co.

AEleen Frisch, *Basic System Administration*, O'Reilly & Associates, Inc.

Intermediate:

David Fiedler and Bruce Hunter, *UNIX System Administration*, Hayden Books.

Advanced:

Russell G. Sage, The Waite Group, *Tricks of the UNIX Masters*, Howard W. Sams and Company.

Donnalyn Frey & Rick Adams, *!%@:: A Directory of Electronic Mail Addressing & Networks, Second Edition*, O'Reilly & Associates, Inc.

Tim O'Reilly & Grace Todino, *Managing UUCP and Usenet*, O'Reilly & Associates, Inc.

SunSoft

A Sun Microsystems, Inc. Business

801-2985-01