

User's Guide

INTERACTIVE™
OPERATING SYSTEM
UNIX®



SunSoft
A Sun Microsystems, Inc. Business

INTERACTIVE™ UNIX® System V/386 Release 3.2 User's Guide

2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

Part No: 801-7871-10
Revision A, June 1994

UNIX is a registered trademark of
UNIX System Laboratories, Inc.,
a wholly owned subsidiary of Novell, Inc.



SunSoft
A Sun Microsystems, Inc. Business

© 1994 Sun Microsystems, Inc.
2550 Garcia Avenue, Mountain View, California 94043-1100 U.S.A.

© 1987-1988 AT&T Corporation

© 1985-1991 INTERACTIVE Systems Corporation

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

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

© 1980-1989 The Regents of the University of California

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

Portions of this product may be derived from the UNIX® and Berkeley 4.3 BSD systems, licensed from UNIX System Laboratories, Inc., a wholly owned subsidiary of Novell, Inc., and the University of California, respectively. Third-party font software in this product is protected by copyright and licensed from Sun's font suppliers.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the United States Government is subject to the restrictions set forth in DFARS 252.227-7013 (c)(1)(ii) and FAR 52.227-19.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

TRADEMARKS

Sun, Sun Microsystems, the Sun logo, SunSoft, the SunSoft logo, Solaris, NFS, ONC, and VP/ix are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and certain other countries. UNIX is a registered trademark of UNIX System Laboratories, Inc., a wholly owned subsidiary of Novell, Inc. DEC and VT220 are trademarks or registered trademarks of Digital Equipment Corporation. Hercules is a trademark of Hercules Computer Technology, Inc. Informix is a registered trademark of Informix Software, Inc. i386, i486, Intel, and Pentium are trademarks or registered trademarks of Intel Corporation. INTERACTIVE and TEN/PLUS are trademarks or registered trademarks of INTERACTIVE Systems Corporation. AIX, AT, IBM, and PC/XT are trademarks or registered trademarks of International Business Machines Corporation. Lotus and 1-2-3 are registered trademarks of Lotus Development Corporation. MS-DOS and XENIX are registered trademarks of Microsoft Corporation. Motif, Open Software Foundation, and OSF are trademarks or registered trademarks of Open Software Foundation, Inc. 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.

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".

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE PUBLICATION. SUN MICROSYSTEMS, INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS PUBLICATION AT ANY TIME.



Please
Recycle

Contents

About This Book	xv
1. Introduction	1
INTERACTIVE UNIX Operating System	2
The UNIX Operating System	2
Enhancements From SunSoft	4
SunSoft Optional Extensions	10
VP/ix Environment	10
INTERACTIVE Software Development System	11
INTERACTIVE TCP/IP	12
INTERACTIVE NFS and NIS Extensions	12
INTERACTIVE X11	13
Enhanced C2 Security Extension	13
<i>Part 1 —INTERACTIVE UNIX System Primer</i>	
2. Getting Started	17
Booting the System	18

Logging In.	19
Your Home Directory	21
Setting a Password (passwd)	21
Changing a Password	22
Logging Out	23
Shutting Down the System	24
3. Using UNIX System Commands	27
Command Syntax	27
Command Names	28
Command Arguments	30
Command Options	30
Common Commands	31
Who Is on the System (who)?	31
What Day Is It (date)?	32
What's in This File (cat)?	33
4. Understanding INTERACTIVE UNIX System Manual Entries	35
5. Interpreting INTERACTIVE UNIX Operating System Messages	39
6. INTERACTIVE UNIX System Directory Structure	43
File and Directory Naming Conventions	45
Path Names	45
Path Name Shortcuts: Dot and Dot Dot	46
Common File and Directory Commands	48
Where Am I (pwd)?	48

Creating a Directory (<code>mkdir</code>).....	49
Changing Directories (<code>cd</code>)	50
What's in This Directory (<code>ls</code>)?.....	51
Copying a File (<code>cp</code>)	52
Removing a File or Directory (<code>rm, rmdir</code>).....	54
7. UNIX System Editors	57
The <code>ed</code> Line Editor.....	57
The <code>vi</code> Screen Editor.....	58
The TEN/PLUS User Interface.....	58
8. Using the Shell	59
What Is a Shell?	59
Shell Features	60
Using Wildcards.....	60
Redirecting Input and Output	65
Using Pipes and Filters.....	67
Background Processing.....	69
9. Other Frequently Used Commands.....	71
Changing Permissions (<code>chmod</code>)	71
Moving a File or Directory (<code>mv</code>).....	76
Changing Owner (<code>chown</code>)	77
Checking Spelling (<code>spell</code>).....	78
Viewing a File (<code>pg</code>).....	79
Printing a File (<code>lpr</code>).....	80
Checking on Processes (<code>ps</code>)	81

Killing a Process (<code>kill</code>)	82
10. Networking Commands	83
Remote Login (<code>rlogin</code>)	85
Remote Copy (<code>rcp</code>)	87
Execute a Shell Command on a Remote Host (<code>rsh</code>)	88
List Users on Remote Hosts (<code>rwho</code>)	90
File Transfer Program (<code>ftp</code>)	91
Remote Login Using Telnet Protocol (<code>telnet</code>)	94
11. Using Virtual Terminals	99
12. Accessing DOS Files	101
Using DOS-FSS	102
Using <code>dossette</code>	105
13. The TEN/PLUS Environment	109
Getting Started	109
Your Home Directory	110
Using ZOOM-IN and ZOOM-OUT	111
Using the Cursor-Positioning Functions	111
Creating Documents	112
Using HELP and CANCEL	116
Creating Directories	116
Editing Files and Directories	117
Using PICK-UP and PUT-DOWN	118
Using PICK-COPY and PUT-COPY	121
Using INSERT and FORMAT	124

Using Menus	128
Using MENU	128
Using LOCAL-MENU	130
More About Editing	132
Changing Margins and Tabs	132
Alternating Between Insert and Overwrite Modes	133
Using +SEARCH, -SEARCH, and BREAK	133
Using USE	134
Printing Documents	134
Summary of TEN/PLUS Functions	135
The Ten Basic Functions of the TEN/PLUS System	135
Additional TEN/PLUS Functions	136
14. Internationalization	141
Background	141
Entering Data	142
U.S. Personal Computer Keyboard Layout	143
Generating Characters Not Present on a U.S. Keyboard	144
European Personal Computer Keyboard Layouts	147
Russian or Greek Keyboards	148
Keyboard Layouts on 7-bit Terminals	149
Using the VP/ix Environment	150
Entering Data and Using INTERACTIVE X11	150
Storing Data in the Computer	151
ASCII	151

8-bit Characters and Codesets	152
IBM Codepages	153
ISO Codesets	155
7-bit Codesets.....	156
Choosing and Configuring a Codeset.....	156
Displaying Data	157
7-bit Terminals	157
The Console	158
Displaying Data and Using INTERACTIVE X11	159
The International Environment	159
Internationalized Behavior	160
Date and Time Format	160
Character Classification	161
Collation	162
Numeric and Monetary Formatting	164
Yes/No Responses.....	164
 <i>Part 2 —System Administration for New Users</i>	
15. System Administration Overview	167
What Is a System Administrator?	167
Before You Begin	168
16. Shutting Down and Bringing Up the System	169
Shutting Down the System	169
Using the powerdown Administrative Login	170
Using the shutdown Command	171

Bringing Up the System	175
17. Special User Accounts on the INTERACTIVE UNIX System	177
System Login Accounts.....	178
The root Login	178
The bin Login	179
Other System Logins.....	179
Administrative Login Accounts.....	180
Accessing Other User Accounts (su).....	181
18. The System Administration Program	183
The sysdemo Login.....	184
Using the CUI Help Facility	185
Hypertext	186
Links	186
Using CUI Menus and Forms.....	187
The Bar Menu.....	187
Pull-Down Menus	187
Forms.....	187
Fields.....	188
Bar Menu Options	189
Using the sysadm Menus.....	191
sysadm Menu Bypass.....	192
19. Managing User Accounts.....	195
What Is a User Login Account?	195
System Files That Define the User's Environment	197

The /etc/passwd File	197
The /etc/group File	198
The /etc/shadow File	199
Managing User Accounts With <code>sysadm</code>	199
Adding a New User	200
Adding a New Group	204
Managing Passwords	205
Aging User Passwords	205
Changing Other Users' Passwords	207
20. Tailoring the System Environment	209
Environment Variables	210
The System Default Profile	212
The User's <code>.profile</code>	213
Setting the System Date and Time	215
Resetting the System Date and Time	217
Setting Up a Message of the Day	218
Changing the News	218
Automatic Program Execution	219
The <code>cron</code> Program	219
Automatic System Cleanup	220
21. sendmail Overview	223
What Does <code>sendmail</code> Do?	223
Sample Mail System Configurations	224
The Location of Mail-Related Files	225

sendmail Installation Instructions	225
Configuring the Mail System Using <code>sysadm</code>	226
22. Understanding INTERACTIVE UNIX System File Systems.	235
What Is an INTERACTIVE UNIX System File System?	235
File System Naming Conventions	238
Mount Point Conventions.	239
Device Naming Conventions	241
Device Naming Conventions for Partitions on Fixed Disks	242
Device Naming Conventions for Diskettes	246
Device Naming Conventions for SCSI Tapes	248
23. Using Utilities With the Diskette Drive	251
Formatting Diskettes	251
Formatting a Diskette Using <code>sysadm</code>	251
Formatting a Diskette From the Command Line	253
Copying Diskettes	253
Copying Files To and From a Diskette	254
Copying Files Using <code>cpio</code>	254
Copying Files Using the <code>tar</code> Command	255
24. File System Maintenance.	257
Creating a File System on a Diskette	257
Mounting a File System.	259
Mounting a Diskette-Based File System	260
Mounting a File System on the Second Fixed Disk	262

Unmounting a File System	264
Unmounting a Diskette-Based File System	264
Unmounting a File System on a Fixed Disk	265
Checking the Space Available on a File System	266
Checking and Repairing a File System	266
File System Corruption and Increasing Reliability	267
Checking a File System	267
Checking a Diskette File System	268
Checking a Fixed Disk File System	269
25. Backing Up Files	271
Before You Back Up Your System	272
Backing Up a File System	272
Backing Up Individual Files and Directories	274
Restoring Files	278
26. Managing Devices	281
What Are Devices and Device Drivers?	281
The Kernel	282
The High Performance Device Driver	282
Tailoring Your System Kernel	283
27. Administering System Security	285
Sources of Potential Damage	285
Basic Precautions	286
System Backups	286
Access Permissions	286

Password Administration.....	288
Glossary	289
Index.....	305

About This Book

This guide provides an overview of INTERACTIVE™ UNIX® System V/386 Release 3.2 from SunSoft. It discusses basic information and procedures for using and administering the INTERACTIVE UNIX Operating System.

Who Should Use This Book

This guide is intended for users of the INTERACTIVE UNIX System who need information about basic tasks, such as creating, copying, and deleting files. The second part of this guide is intended for novice system administrators, or system administrators unfamiliar with the INTERACTIVE UNIX System. It provides step-by-step instructions for common system administration tasks.

Experienced system administrators may refer to the *INTERACTIVE UNIX System V/386 Release 3.2 Maintenance Guide* for more detailed information on most aspects of administering and maintaining the INTERACTIVE UNIX Operating System.

Before You Read This Book

Refer to the *INTERACTIVE UNIX System V/386 Release 3.2 Release Notes* for any last minute caveats, problems, or workarounds.

How This Book Is Organized

This guide is divided into two parts. The chapters in each part are briefly described below.

Chapter 1, "Introduction," provides an introduction to the INTERACTIVE UNIX System Product Family. A brief history of the UNIX Operating System is included, and some of the most important enhancements from SunSoft are described. This chapter also summarizes each optional extension that is available with the INTERACTIVE UNIX Operating System.

Part 1 — INTERACTIVE UNIX System Primer

Chapter 2, "Getting Started," provides basic information about how to get started on the system. It outlines procedures such as how to log in, log out, and set a password.

Chapter 3, "Using UNIX System Commands," explains command syntax, arguments, and options. Basic examples are provided to illustrate how commands are used.

Chapter 4, "Understanding INTERACTIVE UNIX System Manual Entries," explains the format and general content of a manual entry and each reference manual section.

Chapter 5, "Interpreting INTERACTIVE UNIX Operating System Messages," provides general information about how to decipher error messages. Some common error messages are discussed.

Chapter 6, "INTERACTIVE UNIX System Directory Structure," describes and illustrates the hierarchical directory structure, explains file naming conventions, and discusses path names. Additional basic UNIX System commands are also described.

Chapter 7, "UNIX System Editors," describes the differences and advantages of three different UNIX System editors: `ed`, `vi`, and `TEN/PLUS`.

Chapter 8, "Using the Shell," discusses several versions of the shell command interpreter. Shell features such as wildcards, redirection of input and output, pipelines, and background processing are described and illustrated.

Chapter 9, “Other Frequently Used Commands,” describes additional commands that are particularly useful to a user. Moving a file or directory, viewing a file, checking the spelling in a file, and printing a file are among the commands discussed.

Chapter 10, “Networking Commands,” is useful if you have the INTERACTIVE TCP/IP optional extension installed. It covers some basic, user-level networking commands. Topics such as logging in to a remote host, copying files from one host on the network to another, and listing users on a remote host are described.

Chapter 11, “Using Virtual Terminals,” explains what a virtual terminal is and describes how to switch from one virtual terminal to another.

Chapter 12, “Accessing DOS Files,” discusses two mechanisms for accessing MS-DOS® (DOS) files from the INTERACTIVE UNIX System: DOS File System Support (DOS-FSS) and the Dossette File Exchange utility, *dossette*.

Chapter 13, “The TEN/PLUS Environment,” provides instructions on how to use the TEN/PLUS® environment functions. A summary of these functions is provided at the end of the chapter.

Chapter 14, “Internationalization,” explains the internationalization features of the INTERACTIVE UNIX Operating System and describes how the software can be used on computer systems outside the United States. It focuses on usability and is restricted to languages that use an alphabet with fewer than one hundred letters.

Part 2 — System Administration for New Users

Chapter 15, “System Administration Overview,” explains the basic duties of the individual responsible for installing and maintaining a system.

Chapter 16, “Shutting Down and Bringing Up the System,” explains how to boot the system, how to shut it down gracefully, and how to reboot the system.

Chapter 17, “Special User Accounts on the INTERACTIVE UNIX System,” discusses various privileged users and how to access their accounts.

Chapter 18, “The System Administration Program,” *sysadm*, describes the system administration program available with this system and describes how to access its different levels.

Chapter 19, “Managing User Accounts,” describes the attributes of each user login and explains how to use the `sysadm` command to add, change, or delete user accounts.

Chapter 20, “Tailoring the System Environment,” describes the environment variables, the most commonly configured system files, and the `cron` program. It also explains how to set the date and time and change the message of the day.

Chapter 21, “sendmail Overview,” provides an overview of how to configure the `sendmail` internetwork mail routing facility.

Chapter 22, “Understanding INTERACTIVE UNIX System File Systems,” explains the structure of INTERACTIVE UNIX System file systems and describes the conventions used to name file systems and devices.

Chapter 23, “Using Utilities With the Diskette Drive,” explains how to format diskettes, copy diskettes, and copy files to diskettes.

Chapter 24, “File System Maintenance,” explains how to create and maintain INTERACTIVE UNIX System file systems.

Chapter 25, “Backing Up Files,” describes how to back up and restore files.

Chapter 26, “Managing Devices,” gives a brief explanation of devices, device drivers, the kernel, and the High Performance Device Driver.

Chapter 27, “Administering System Security,” discusses general requirements for system security and presents basic precautions that should be taken.

Glossary is a list of words and phrases found in this book and their definitions.

Related Books

The following books discuss advanced system administration tasks, and installation of the INTERACTIVE UNIX System and related packages.

- *INTERACTIVE UNIX System V/386 Release 3.2 Maintenance Guide*
- *INTERACTIVE UNIX System V/386 Release 3.2 Installation Guide*

Conventions

Various font types are used in this guide to distinguish between information you type, information displayed by the system, and items that either you or the system replace with a variable. Table P-1 shows the type changes and symbols used in this book.

Table P-1

Typeface or Symbol	Meaning	Example
AaBbCc123	An initial capitalized key name refers to that key on your keyboard	Press Enter to store the file. Press Control-d to quit.
AABBCC123	An all capitalized key name refers to a TEN/PLUS function	ZOOM-IN to the file.
AaBbCc123	The names of commands, files, and directories; on-screen computer output	Edit your .profile file. Use <code>ls -a</code> to list all files. \$ You have mail.
AaBbCc123	What you type, contrasted with on-screen computer output	\$ rlogin host2 password:
<i>AaBbCc123</i>	Placeholder: replace with a real name or value; book titles; first mention of a new term that is in the glossary; words to be emphasized	To delete a file, type <code>rm filename</code> . See the <i>INTERACTIVE UNIX System User's Guide</i> . The system displays a <i>prompt</i> . You <i>must</i> be root to do this.

Code samples are included in plain boxes and may display the following:

UNIX C shell prompt	%
UNIX Bourne and Korn shell prompt	\$
UNIX superuser prompt	#

Note – Keys on your keyboard may be labeled differently than those shown in this guide. For example, the Enter key is labeled Return on some keyboards. If your hardware or software vendor supplies additional documentation with your system, read that documentation for information on key names before you continue.

Plain boxes represent screen displays, system responses, file contents, path names, or program code. They may contain text that you type, which is indicated by bold lettering. For example:

```
login: tony
Password:
INTERACTIVE UNIX System V/386 Release 3.2, Version 4.1
amadeus
Copyright (C) 1994 Sun Microsystems, Inc.
Copyright (C) 1988 AT&T
Copyright (C) 1988 Microsoft Corp.
All Rights Reserved
Login last used: Fri May 13 16:59:30 1994
```

In Part 1 — “INTERACTIVE UNIX System Primer,” new commands are introduced in a double-boxed table. This display provides basic information about the command’s format (usage), description, options, and arguments.

Highlighting in screens is represented by reverse type. For example:

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

```
SYSTEM ADMINISTRATION
```

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. This is explained further in Chapter 4, “Understanding INTERACTIVE UNIX System Manual Entries.”

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

Full Title	Shortened Version
<i>INTERACTIVE UNIX System V/386 Release 3.2 User's Guide</i>	<i>INTERACTIVE UNIX System User's Guide</i>
<i>INTERACTIVE UNIX System V/386 Release 3.2 Installation Guide</i>	<i>INTERACTIVE UNIX System Installation Guide</i>
<i>INTERACTIVE UNIX System V/386 Release 3.2 Maintenance Guide</i>	<i>INTERACTIVE UNIX System Maintenance Guide</i>
<i>INTERACTIVE UNIX System V/386 Release 3.2 Release Notes</i>	<i>INTERACTIVE UNIX System Release Notes</i>
<i>INTERACTIVE UNIX System V/386 Release 3.2 Conformance Guide</i>	<i>INTERACTIVE UNIX System Conformance Guide</i>
<i>INTERACTIVE UNIX System V/386 Release 3.2 Hardware Compatibility List</i>	<i>INTERACTIVE UNIX System Hardware Compatibility List</i>

Note that in documentation prepared by UNIX System Laboratories (USL), references to certain USL books now refer to information present in different SunSoft documents.

USL Document	SunSoft Document
<i>UNIX System V/386 Release 3.2 Operations/System Administration Guide</i>	<i>INTERACTIVE UNIX System User's Guide</i>
	<i>INTERACTIVE UNIX System Maintenance Guide</i>
<i>UNIX System V/386 Release 3.2 Release Notes</i>	<i>INTERACTIVE UNIX System Release Notes</i>

The INTERACTIVE UNIX Operating System 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 and DOS applications under the INTERACTIVE UNIX System. Other SunSoft extensions provide users with powerful graphics 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 chapter provides an overview of the INTERACTIVE UNIX System Product Family line, which includes:

INTERACTIVE UNIX Operating System

Provides an overview of the INTERACTIVE UNIX Operating System and describes the UNIX System features and enhancements from SunSoft.

SunSoft Optional Extensions

Describes the optional programming, networking, graphical, and other extensions that are available to accompany the INTERACTIVE UNIX System.

INTERACTIVE UNIX Operating System

The INTERACTIVE UNIX System is comprised of:

- The UNIX Operating System
- Enhancements from SunSoft

The INTERACTIVE UNIX Operating System is an enhanced version of the UNIX Operating System and is the foundation of the INTERACTIVE UNIX System 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 wholly owned subsidiary of Novell, Inc., 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 x86-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 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, the 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 provided for XENIX V/386 system call extensions and several XENIX commands to enhance compatibility.

Internationalization includes support for 8-bit codesets. Commands such as `cat`, `vi`, `grep`, and `ls` handle codesets 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 local language and format.

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, IPX/SPX, 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 Prentice-Hall. There are also a number of excellent books commercially available that describe the UNIX Operating System and its components.

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 x86-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 can be done by 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 installing and removing software, formatting diskettes and the fixed disk, backing up the fixed disk, setting and changing system passwords, and setting up the system to use modems, UUCP, and networking.

Documentation

SunSoft supplies a set of documentation tailored specifically to the INTERACTIVE UNIX Operating System that compliments the traditional set of UNIX System documentation. The manual entries are delivered as an installable subset.

Kernel Configuration Link Kit

The INTERACTIVE UNIX System Kernel Configuration Link Kit 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 and tape 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 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 full capabilities of supported peripheral controllers.
- To support a larger number of controllers.
- To make it easy to add support for new kinds of controllers, disks, and related 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 the INTERACTIVE UNIX Operating System, SunSoft provides controller-specific HPDD disk modules for a variety of ST-506, RLL, ESDI, IDE, MFM, and SCSI controllers for machines with ISA, EISA, PCI, VL, 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 the Berkeley UFS file system, which gain speed by completely restructuring the file system, thus making them incompatible with the standard System V (s5) file system.

The Fast File System works only with the standard System V 1K-block (S51K) file system and the SunSoft-enhanced S5L 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 using the FFS than with the 2K-block file system for most applications. (The S5L ("L" for long) file system is the same as the S51K file system, except that it supports file names of up to 512 characters.)

Enhanced File System Layout

The file system layout is designed to contain all UNIX System-specific structures within 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:

- **dosette 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 standard UNIX System file and record locking functions.

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 (84, 101, and 102 keys)
- Display adapters (monochrome, CGA, EGA, VGA, SVGA, and Hercules™)
- Standard serial ports (COM1, COM2, etc.)
- Diskettes (5.25- and 3.5-inch media)
- Fixed disks (RLL, SCSI, IDE, MFM, and ESDI)
- Parallel printers
- Streaming tapes (using both dedicated controllers and SCSI controllers)
- CD-ROM devices

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” switch between active processes. This facility supports up to eight concurrent virtual terminal sessions on the system console and supports graphics applications.

Selectable Console

This feature allows any serial terminal on the system (not just the PC monitor) to act as the console. This makes it possible to do system administration tasks, including rebooting the system, from a remote site.

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 using UNIX-to-UNIX 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.

Korn Shell

This increasingly popular command interpreter combines many features of the Bourne shell and the capabilities of the C shell.

The TEN/PLUS Environment

The TEN/PLUS User Interface is designed to make the system easy to learn and easy to use. It contains a powerful editor with consistent functions. The same command will have the same result, whether you are reading a mail message, writing a computer program, or updating your calendar. You don't have to learn a new set of commands each time you use a new application or try a new task.

The TEN/PLUS Environment includes the TEN/PLUS Mail System, an electronic message system that provides each user with a private mailbox. It allows you 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, functions keys, and on-line help. It interfaces with Berkeley `sendmail`, UNIX System V `mail`, and other mail systems.

Internationalization

The INTERACTIVE UNIX System contains a number of proprietary utilities and improvements supporting internationalization. The `ttymap` utility can be used to remap characters to properly support different keyboards and terminals. It supports any 8-bit codeset 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 codepages, 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)`, `ttymap(1)`, and `loadfont(5)` for details.

POSIX and XPG Compliance

The INTERACTIVE UNIX Operating System conforms to a number of operating system standards:

- The traditional SVID standard conformance adopted by System V.
- The POSIX/FIPS standard as specified by *IEEE Std. 1003.1-1990* and FIPS 151-2.
- The XPG standard as specified by the *X/Open Portability Guide*. This standard is built on POSIX 1003.1 and contains extensions useful for internationalizing applications.

The traditional SVID environment and the POSIX/FIPS/XPG environment are supported concurrently. The choice between environments is made at compilation time; only programs compiled for POSIX/FIPS/XPG will execute with that environment. For information about standards conformance and internationalized environments, refer to the *INTERACTIVE UNIX System Conformance Guide*, available electronically. As standards evolve (for example, the next release of XPG, Spec 1170), the product will be updated to comply with more recent versions.

SunSoft Optional Extensions

SunSoft provides a variety of extensions designed to further enhance the power of your x86-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 the INTERACTIVE UNIX System. 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, combined with the Integrated DOS File System Support, 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.

INTERACTIVE Software Development System

The INTERACTIVE Software Development System (SDS) contains all the necessary tools to develop applications for the INTERACTIVE UNIX System. It includes all the standard UNIX System V/386 Release 3.2 utilities for developing and maintaining software programs, including USL's portable C compiler, the Source Code Control System (SCCS), `make`, `lex`, `yacc`, etc. It also contains several libraries, including the standard C library (shared and non-shared), the Extended Terminal Interface (`curses` library, etc.), as well as the libraries for the X Window System (the X library, the X Toolkit library, and the Athena widgets). Manual pages for all but the X11 functions are provided on-line as well. Using the special POSIX library and a compiler flag, it is possible to create POSIX 1003.1 conformant applications (without this flag, standard UNIX System V.3.2 applications are created).

The Optimizing C Compiler for the INTERACTIVE UNIX System is bundled in with the INTERACTIVE UNIX System.

Optimizing C Compiler for the INTERACTIVE UNIX System

The Optimizing C Compiler for the INTERACTIVE UNIX System provides highly-optimized translation of CPU-intensive programs targeting i486™ and Pentium™ processors. It can perform classical, memory, and interprocedural optimizations and branch-count profiling to improve program performance. The compiler produces fast running binaries for a variety of Intel processors, including the i386™; special compiler options produce further optimization when the binary is targeted exclusively at either the i486 or Pentium processor.

The compiler's implementation of C conforms to the ANSI Standard for the C language. It also supports several extensions to the ANSI C language as well as standard K&R C. A highly optimized version of the library of mathematical functions is supplied with the compiler, as well as a Compiler Guide.

Software Integration Tools

This subset contains easy-to-use tools that can be used with application software to build a subset that can be installed with the `sysadm` facility.

INTERACTIVE X11 Development System

The INTERACTIVE X11 Development System includes libraries 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 TCP/IP

INTERACTIVE TCP/IP facilities provide the standard Transport Control Protocol/Interface Program (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 other networking commands. 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 and NIS Extensions

INTERACTIVE NFS 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 different 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.

The Network Information Service (NIS) is part of SunSoft's ONC[™] distributed services. The ONC environment is made up of a core platform of interprocess communication protocols, on which a variety of services are built. The best known of these is the NFS system. NIS provides a network-wide data management facility. It provides an extensible database for storing system information such as host names, network addresses, user names, and networks.

INTERACTIVE X11

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

INTERACTIVE X11 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.

Enhanced C2 Security Extension

The Enhanced C2 Security Extension is an optional extension to the INTERACTIVE UNIX System from SunSoft 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). It is a usable and easily administered software package 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 Enhanced C2 Security Extension corrects, modifies, and adds to the standard utilities and operating system to define its TCB.

Part 1 — INTERACTIVE UNIX System Primer

This primer is an introduction to the basic components and features of the INTERACTIVE UNIX Operating System. It covers such topics as using commands, working with files and directories, and interpreting error messages.

This primer provides you with the information you need to start using the INTERACTIVE UNIX Operating System. It is not intended as a technical reference manual. If you require more information, refer to the *INTERACTIVE UNIX System Maintenance Guide*.

Note – If you are a new UNIX System user, it is important that you read this primer in its entirety as it presents commands and steps in the usual order in which you will need them.

Using this primer, you will learn about:

- Executing and stopping INTERACTIVE UNIX System commands
- Interpreting system error messages
- Creating directories to organize documents such as letters and reports
- Using the command interpreter (shell)

If your system includes INTERACTIVE TCP/IP, see Chapter 10, “Networking Commands.” It covers some basic, user-level networking commands and applications, such as:

- Copying files and directories from one host on the network to another
- Logging in to a remote host
- Executing a shell command on a remote host
- Listing users on a remote host
- Finding out the status of a remote host

To reinforce what you read in this primer, try each example on your system. Remember that new terms introduced in *italics* can be found in the glossary at the end of this guide.

Before you can begin working on the system, you or your system administrator must install your system and set up your *login account*. If the power to your computer is off, you must *boot* (bring up) the system before you can *log in*.

Booting the System

Before you can log in to your computer, you must boot the *operating system*. The operating system is the group of programs and utilities that manages the computer's resources, processes user requests, and runs application programs. You must boot the system at the *console*, or main terminal, which is usually the terminal directly attached to the computer unit.

Note – A serial terminal may be configured as the console. Refer to the *INTERACTIVE UNIX System Maintenance Guide* and *chgcon(1M)*.

To boot the system:

1. Make sure there is no diskette in the diskette drive.
2. Do one of the following:
 - If you have turned off your computer, turn it on again. The INTERACTIVE UNIX Operating System is automatically booted from the fixed disk.
 - If the power to your computer is turned on and your system has been previously shut down (refer to “Shutting Down the System” in this chapter), press any key to reboot your system.
3. Booting the INTERACTIVE UNIX System appears on your screen and the computer begins to boot. The screen then clears and a number of messages appear. The system is ready for you to log in when you see the message `Console Login: .`

Logging In

To use the INTERACTIVE UNIX System:

1. Turn on your computer or terminal and log in.
 - If you are logging in to the console terminal, your screen will look like this:

```
Console Login:
```

- If you are using a terminal other than the console terminal, your screen will look similar to this:

```
login:
```

2. Type the user identification name (ID) assigned to you (this is also called your login name), then press Enter. Your user ID can contain a maximum of eight characters and can consist of uppercase letters, lowercase letters, and numerals. By convention, most user IDs on UNIX Systems consist of lowercase letters.

3. After a user ID is entered, the system may request a password:

```
Console Login: tony  
Password:
```

- If you have been assigned a password by your system administrator, when Password: displays, type your password and press Enter.

Note – The system does not display the password you type on the screen.

- If you have not been assigned a password, your screen will look similar to this:

```
Console Login: tony
INTERACTIVE UNIX System V/386 Release 3.2, Version 4.1
amadeus
Copyright (C) 1994 Sun Microsystems, Inc.
Copyright (C) 1988 AT&T
Copyright (C) 1988 Microsoft Corp.
All Rights Reserved
Login last used: Fri May 13 16:59:30 1994
$
```

If you accidentally log in using uppercase letters, the system assumes that your terminal (even the system console) is not capable of handling lowercase letters. All input and output for the remainder of the session will be shown in uppercase letters. If this happens, log out (type `exit`), make sure your Caps-Lock key is not engaged, and log in again.

When you have completed the login procedure, the system displays a *prompt* on the screen. The prompt is a symbol, usually a dollar sign (\$) or a percent sign (%), but because this can be changed by your system administrator, your prompt may be different. The prompt indicates that the system is ready to receive information. You may enter a *command* or run an *application* when the prompt is displayed on your screen. A command is an instruction you give the computer, and an application is a computer program that helps you perform a certain type of work.

When you interact with the operating system, the commands you type are processed by a *command interpreter*, which passes your commands to the operating system for processing and delivers the results to you. The INTERACTIVE UNIX System command interpreter is called the *shell*, and the prompt that is displayed is usually called the *shell prompt*. You will learn more about the shell in Chapter 8, "Using the Shell."

Your Home Directory

Each time you log in, the system places you in your *home directory*. Your home directory serves as your personal work area.

A *directory* (such as your home directory) can contain both files and other directories. It is similar in function to a file cabinet: a file cabinet has several drawers (directories), each of which can contain many folders (subdirectories), each holding one or more documents (files). A *file* is a collection of data stored under an assigned name. For example, a letter, a sales report, or payroll data can all be stored in files. You will learn more about directories and files in Chapter 6, "INTERACTIVE UNIX System Directory Structure."

Setting a Password (`passwd`)

If you were not assigned a password when your login account was set up, the system will probably request that you choose one the first time you log in. A password keeps unauthorized users from tampering with your files by assuring that no one can log in to your account on the computer. Once your password is set, only you, or someone who knows your password, can access your account. (See "Command Syntax" in Chapter 3, "Using UNIX System Commands" for an explanation of the command format shown below.)

COMMAND NAME	<code>passwd</code>
FORMAT	<code>passwd</code>
DESCRIPTION	Set or change your login password. The program prompts for the old password (if any) and prompts twice for the new password.
OPTIONS	None.
ARGUMENTS	None.

1. Log in to the system.

- The system may prompt you immediately for a new password. Continue with the next step.

```
login: tony
You don't have a password. Choose one.
passwd tony
New password:
```

- If the system does not prompt for a new password after you log in, refer to the next section, “Changing a Password.”
2. Enter a term between six and eight characters long, then press Enter. Your password should be one that others can't readily figure out but that will be easy for you to remember. It must contain at least one numeric or special character. Special characters include the space character and:

```
< > * ? | & $ ; \ " ' ^ ( ) [ ]
```

3. When you type in the password you have chosen, the system does not display it on the screen. The system verifies the password by asking you to type it again:

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

If you make a mistake entering your password the second time, the system will give you another chance.

Changing a Password

Some system administrators assign all new users a default password when their login accounts are established. You should change this as soon as you log in so that others who may know the default password do not have access to your account. Use the `passwd` command to change your current password.

1. Type the `passwd` command at the shell prompt, then press Enter:

```
$ passwd
```

Note – Remember to type the command exactly as it is shown in the example. You cannot type `password` or any other variation of the command.

2. The system asks you to type in your old password, prompts you for a new one, and then has you verify the new one by typing it a second time:

```
$ passwd
passwd: Changing password for tony
Old password:
New password:
Re-enter new password:
```

Note – Don't forget your password! You need it to log in to the system.

Logging Out

To end your access to the system, you must *log out*. Always log out when you have finished using the computer, to prevent unauthorized use of your account.

On most systems there are two ways you can log out.

- You can hold down the Control key and simultaneously press d.
- Or you can type the `exit` command at the prompt:

```
$ exit
```

It is not necessary to press Enter after logging out. The computer will display the `login:` prompt, indicating that it is ready to accept a login name for the next session. If you are using a dial-in (modem) line, the system will hang up the connection when you log out.

Shutting Down the System

The INTERACTIVE UNIX Operating System is a *multi-tasking* system. A multi-tasking system means that the computer can run many different processes (programs) at the same time. For example, you can edit a file at the same time another file is being printed on your printer. When you want to turn off your computer, you must make sure that all the tasks currently running are stopped in an orderly fashion. Do this by logging in with `powerdown`, a special administrative login account. `powerdown` runs a system maintenance procedure called `shutdown`, a program that gracefully terminates the tasks that are currently executing before halting the system. When `shutdown` has finished, you can safely turn off the computer.



Caution – If you do not run the shutdown program, you may lose data that is stored on your system and cause damage to your *file system*.

You must be authorized to use the `powerdown` account. To shut down the system:

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

Note – You must know the `powerdown` password if one has been set (see your system administrator).

3. Once you have completed the login procedure using the powerdown login, the system automatically executes the shutdown program. The system displays a screen similar to this:

```
login: powerdown
Password:
INTERACTIVE UNIX System V/386 Release 3.2, Version 4.1
amadeus
Copyright (C) 1994 Sun Microsystems, Inc.
Copyright (C) 1988 AT&T
Copyright (C) 1988 Microsoft Corp.
All Rights Reserved
Login last used: Thu Apr  7 20:31:37 1994
/           :           Disk space:   82.84 MB of 100.00 MB available(82.84%)
/home      :           Disk space:  222.46 MB of 226.00 MB available(98.44%)

Total Disk Space:                305.30 MB of 326.00 MB available(93.65%)

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.   Thu Apr  7 20:33:37 PDT 1994

Broadcast message from root (console) on amadeus Thu Apr  7 20:33:38...
THE 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.
cron aborted: SIGTERM
! SIGTERM Thu Apr  7 20:33:43 1994
! ***** CRON ABORTED ***** Thu Apr  7 20:33:43 1994

The system is down.
Press any key to reboot.
```


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

On the INTERACTIVE UNIX Operating System, the words *command* and *program* are nearly synonymous. In simple terms, the user types a command and presses Enter, and the INTERACTIVE UNIX System runs the program that performs the user's command.

An INTERACTIVE UNIX System *command line* can have three parts: the *command* name, its *options*, and its *arguments*. Each command line is entered in to the computer by pressing Enter. When the Enter command is received, the shell sends it to the operating system for execution.

To stop the execution of a command, press Delete. (Your system administrator may have changed the interrupt key to Control-c.)

Command Syntax

The INTERACTIVE UNIX System is *case sensitive*, which means that the system distinguishes between uppercase and lowercase letters. Most INTERACTIVE UNIX System commands, options, and arguments are typed in lowercase letters. Options typically begin with a dash (-). Each command, option, or argument consists of one word, which is interpreted as a group, or *string*, of characters surrounded by spaces.

If you make an error when typing a command, use the Backspace key to correct the error. You *cannot* use the *cursor positioning* (arrow →) keys.

Always type the command name first, followed by a space; the desired option or options, each followed by a space; then any arguments, separated by spaces.

Here is the command format:

COMMAND NAME	command name
FORMAT	command [<i>option(s)</i>] <i>argument(s)</i>
DESCRIPTION	A brief description of what the command does.
OPTIONS	A list of the most useful options and a brief description of each.
ARGUMENTS	Mandatory or optional arguments.

If an argument is not required, it is shown in square brackets []. Options are always “optional,” so they are always shown in square brackets []. Only the most common options and arguments are discussed in this primer. If there are additional options or arguments available for a particular command that are not presented in this primer, this is indicated by the phrase “Not presented in this document.” For a complete listing of the available options and arguments for a command, refer to the reference manuals for your system.

Command Names

Most INTERACTIVE UNIX System command names are short or abbreviated words that describe the programs they invoke. They are deliberately kept short to save time and reduce typing errors. The `passwd` command is an example of an abbreviated word used as a command name. When you enter an INTERACTIVE UNIX System command name, do *not* abbreviate it or change its spelling in any way, or the command won’t work.

For the novice user, INTERACTIVE UNIX System command names can be cryptic and confusing. Because the UNIX System was originally developed for systems programmers, many of the commands were not created with the new user in mind. However, you need only learn a few commands to run application programs and other facilities on your system. If you use a user-friendly interface, such as the TEN/PLUS interface, you may not have to learn many UNIX System commands at all.

Using a Simple Command

You have already used the `passwd` command to set or change your system password. `passwd` is an example of a simple command that requires only the command name to execute properly.

The `ls` command can also be executed using just the command name. `ls` is used to “list” the names of the files and directories in the current directory. (You will learn more about files and directories in Chapter 6, “INTERACTIVE UNIX System Directory Structure.”)

COMMAND NAME	<code>ls</code>
FORMAT	<code>ls [file name(s)]</code> <code>ls [directory name(s)]</code>
DESCRIPTION	For each directory named, alphabetically list its contents. For each file named, list the requested information.
OPTIONS	See “What’s in This Directory (ls)?” in Chapter 6, “INTERACTIVE UNIX System Directory Structure.”
ARGUMENTS	A file name or a directory name. If no file or directory is named, list the current directory.

If you type `ls` at the shell prompt, the system will display an alphabetical listing of the file and directory names in the *current working directory*. For example:

```
$ ls
bin
bag.acct
gen.memo
letters
memos
midwest
notes
north
south
```

If your current directory is empty, nothing will happen except that the system displays the shell prompt again:

```
$ ls
$
```

Command Arguments

An argument gives the system information that is required to process a specific command or to change the *default*, or standard, behavior of a command. Some commands require one or more arguments; for others, arguments are optional. An argument usually consists of a file, directory, or user name. Refer to the reference manuals for your system to determine the requirements of each command.

`ls` is an example of a command that accepts a file or directory name as an optional argument. If you type `ls` followed by a file name, the system redisplay the file name if the file exists in the current directory:

```
$ ls notes
notes
```

If a directory name is used as the argument, the system displays the names of all the files and directories contained in that directory:

```
$ ls letters
fox.ltr
lit.ltr
```

Command Options

An option (sometimes called a *flag*) is a special kind of argument that is specific to a particular command. As its name implies, an option is not required, but it provides additional versatility when used with a command. Most options are preceded by a dash (-). Some UNIX System commands can accept both arguments and options.

The `ls` command has several options. The `-l` or “long form” option is illustrated here. If you type `ls -l`, the system lists not only the names of the files and directories in the current directory, but also the owner, file access protections, size in bytes, and time of last modification for each file and directory (some of these file attributes are discussed in Chapter 9, “Other Frequently Used Commands”). A typical directory listing looks like this:

```
$ ls -l
drwxr-xr-x 3 tony other 69979 Apr  2 14:41 bin
-r--r--r-- 1 bobr other 30873 Apr 16 09:02 bag.acct
-rwxrw-r-- 1 tony other 2 Apr  7 13:24 gen.memo
drwx----- 2 tony other 6107 Apr  2 10:48 letters
-rw-r--r-- 1 tony other 52709 Apr  8 16:41 memos
-rw----- 1 tony other 1338 Apr 16 13:04 midwest
-rw-rw-rw- 1 tony other 34395 Apr 20 11:43 notes
-rw----- 1 tony other 44978 Apr 17 16:32 north
-rw----- 1 tony other 922 Apr 17 11:21 south
```

You will learn more about the output of this command in later chapters.

Common Commands

Who Is on the System (who)?

After you have logged in, you can use the `who` command to see a list of all users who are currently logged in.

COMMAND NAME	<code>who</code>
FORMAT	<code>who [am i]</code>
DESCRIPTION	List the users logged in to the system.
OPTIONS	Not presented in this document.
ARGUMENTS	List the information for the user only.

The system displays the name, the terminal ID, the date, and time of login for each user logged in to the system:

```
$ who
barbara  tty04      Mar 15 08:44
michael  console     Mar 15 07:23
terryr   ttyd2       Mar 15 09:02
```

Use `who` with the arguments `am i` to determine the login information for the terminal:

```
$ who am i
tony     console     Mar 15 07:23
```

What Day Is It (date)?

You can use the `date` command to find out the date and time.

COMMAND NAME	<code>date</code>
FORMAT	<code>date</code>
DESCRIPTION	List the current date and time.
OPTIONS	Not presented in this document.
ARGUMENTS	Not presented in this document.

The `date` command displays output similar to this:

```
$ date
Wed Mar 23 11:53:14 EST 1994
```

What's in This File (cat)?

The `cat` command is used to display the contents of a file on the terminal screen.

COMMAND NAME	<code>cat</code>
FORMAT	<code>cat [-s] [file name(s)]</code>
DESCRIPTION	Print one or more files on your terminal screen. If more than one file name is supplied, it is “concatenated” to (added onto the bottom of) the previous file and printed.
OPTIONS	<code>-s</code> Be silent. Do not comment about files that do not exist.
ARGUMENTS	The file name(s) in the order that you want them to print.

For example, to view the contents of the file `/etc/passwd` (the system file that stores information about each user ID on the system), type:

```
$ cat /etc/passwd
```

The system displays the contents of the file on your screen. If the file is a long one, the system will scroll the contents on your screen. To initiate a pause in the scrolling, hold down the Control key and simultaneously type `s`. To resume scrolling, hold down the Control key and simultaneously type `q`. To display the contents of two files, use the `cat` command followed by the two file names, which are separated by spaces:

```
$ cat /etc/passwd /etc/group
```

In the previous example, the file `/etc/passwd` is displayed, followed by the file `/etc/group`.

Understanding INTERACTIVE UNIX System Manual Entries



One type of documentation common to virtually all UNIX-based Systems is the on-line reference manual, which consists of *manual entries*. Manual entries (man pages) follow a specific format, which is briefly explained in this chapter.

The manual entries supplied with the INTERACTIVE UNIX System contain information about the standard UNIX System Laboratories, Inc. (USL) commands, as well as descriptions of enhancements made by SunSoft. See the *INTERACTIVE UNIX System Release Notes* for more up-to-date information.

Manual entries are grouped alphabetically within the following sections:

- Section 1 – Commands and Application Programs
- Section 2 – System Calls
- Section 3 – Subroutines
- Section 4 – File Formats
- Section 5 – Miscellaneous Facilities
- Section 6 – Games
- Section 7 – Special Files
- Section 8 – System Maintenance Programs and Procedures

While all manual entries follow a similar format, only the Section 1 commands are discussed here, since as a new user you are most likely to use the data provided there. Entries in other sections follow the same general rules.

Every entry contains certain headings. A Section 1 entry *always* has a name line, a synopsis, and a description. Other sections are added as needed.

Following is a typical manual entry for a simple user command, `cat`.

CAT(1)

CAT(1)

NAME

`cat` – concatenate and print files

SYNOPSIS

`cat [-u] [-s] [-v [-t] [-e]] file . . .`

DESCRIPTION

cat reads each *file* in sequence and writes it on the standard output.

Thus:

`"cat file"`

prints **file** on your terminal screen, and

`cat file1 file2 >file3`

concatenates **file1** and **file2** and writes the result in **file3**.

If no input file is given, or if the argument `-` is encountered, *cat* reads from the standard input file.

The following options apply to *cat*:

- u** The output is not buffered. (The default is buffered output.)
- s** *cat* is silent about nonexistent files.
- v** Causes nonprinting characters (with the exception of tabs, new-lines, and form-feeds) to be printed visibly. ASCII control characters (octal 000 – 037) are printed as `^n`, where *n* is the corresponding ASCII character in the range octal 100 – 137 (`@`, `A`, `B`, `C`, . . . , `X`, `Y`, `Z`, `[`, `\`, `]`, `^`, and `_`); the DEL character (octal 0177) is printed `^?`. Other nonprintable characters are printed as `M-x`, where *x* is the ASCII character specified by the low-order seven bits.

The following options may be used with the `-v` option:

- t** Causes tabs to be printed as `^I`'s and form feeds to be printed as `^L`'s.
- e** Causes a `$` character to be printed at the end of each line (prior to the new-line).

The `-t` and `-e` options are ignored if the `-v` option is not specified.

WARNING

Redirecting the output of *cat* onto one of the files being read will cause the loss of the data originally in the file from being read. For example, typing:

`cat file1 file2 >file1`

causes the original data in **file1** to be lost.

SEE ALSO

`cp(1)`, `pg(1)`, `pr(1)`.

To read a manual entry, type `man entry name` to see the text of the entry on your screen. The following explains some of the more common sections that comprise a manual entry:

Name line

The name line contains the name of the command and a brief description of its function.

Synopsis line

The synopsis is a line (or set of lines, if the command is more complicated) that summarizes how the command is stated at the system prompt with its options and arguments. The command is shown in bold face; options are bold and are always shown inside square brackets `[]`. *file* (a file name) is the argument needed for the command, and the three dots “...” mean that you can give more than one file name at a time as arguments.

Description section

The description is just that—a description of how the command and its options function. For example, in `cat(1)` the description begins, “*cat* reads each *file* in sequence and writes it on the standard output.” (The standard output device is your terminal screen. This is discussed in Chapter 8, “Using the Shell.”) It then shows you how to “*cat*” more than one file at a time and direct the results into a third file. The rest of the description gives details about using various options.

See Also section

This section contains cross-references to other manual entries or documents that contain information of related interest.

Miscellaneous sections

Some other sections found in various manual entries include the *Warnings* or *Caveats* section, which alerts you to possible problems when using the command in certain ways; the *Examples* section, which contains examples using the more complicated commands; the *Files* section, which contains the system files in which the command is located; the *Diagnostics* section, which contains error messages of interest to programmers who may be using the command in a program; the *Bugs* section, which alerts you to known problems with the command; and the *Notes* section, which contains helpful hints or information that doesn’t fall into another section.

Interpreting INTERACTIVE UNIX Operating System Messages

5 

If something goes wrong when you use an INTERACTIVE UNIX System command, the system usually generates an *error message* to help you determine why the command did not work. Error messages are typically generated when you misspell a command or file name, when a command cannot be automatically accessed from your account, or when a file or directory name does not exist. Error messages on the INTERACTIVE UNIX System can be terse and cryptic. However, as you gain experience with the system, you will become more adept at deciphering them.

Note – Commands and permissions mentioned in this chapter are explained later in this primer.

If you spell a command or file name incorrectly, an error message is displayed on your screen. For example, if you type `datte` instead of `date`, the system displays this error message:

```
$ datte
datte: not found
```

When using the C shell (`csh`), the following error message is displayed:

```
% datte
datte: Command not found.
```

The INTERACTIVE UNIX System also generates an error message if you fail to use a space to separate the command name from its arguments. For example, if you type `ls-l` instead of `ls -l`, the system generates the same message:

```
$ ls-l
ls-l: not found
```

Some common error messages are:

command name: not found

You may have misspelled the command or forgotten a space that was needed. Check your spelling and try typing the command again. If the problem persists, then either your search path does not include the directory where the command is stored or the command is not available on your system. Check with your system administrator.

directory name: bad directory

You may have misspelled a directory name, but this message also appears if the directory does not exist. Check your spelling or change directories (`cd`) to the parent of the directory you are trying to reach and verify that the directory exists and that you have the correct spelling of its name. If you get this message while trying to `cd` into a directory, it may be that you do not have execute permission for that directory. Check the permissions for the directory by using the `ls -l` command while in the parent directory.

Invalid argument

You have used an argument that the command does not accept. Try typing the name of the command without an argument. The system will often display a message that describes how to use the command. You should also check your command reference manual for the correct arguments.

No such file or directory

You have specified the name of a file or directory that does not exist. Often this means that one of the names in your path name is misspelled. Check your spelling and try again. If the problem persists, `cd` to the parent of the directory you are trying to reach and verify that the directory exists and that you have the correct spelling of its name.

Permission denied

You do not have permission to write in the named directory or to perform some operation on the named file. Check the permissions of the target with `ls -l`. Change them or have the owner of the file or directory change them, as necessary.

`cp: Insufficient arguments (1)`

Usage: `cp f1 f2`

`cp f1 ... fn d1`

You have failed to specify the destination file name while using `cp`. This type of message shows the correct usage.

INTERACTIVE UNIX System

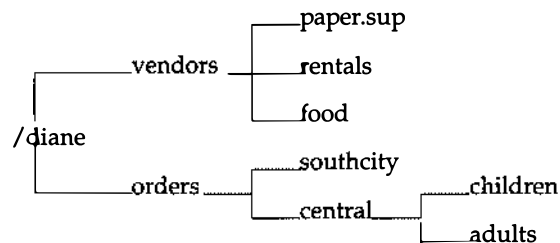
Directory Structure

6 

On an INTERACTIVE UNIX Operating System, information is stored in an organized structure, called a *hierarchical file system*. It is called hierarchical because of its multi-level structure. A single master directory is at the top level, and additional files and directories are defined at various levels below it.

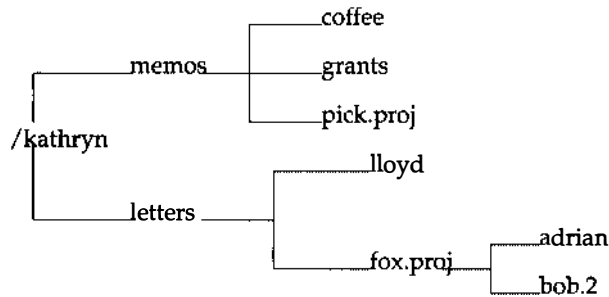
A *file system* is a collection of individual files and directories that are stored on a portion of a disk.

The master directory for the entire system is called the *root directory*. It is the first directory in the file system structure and is named slash (/). The root directory contains several directories, and each of these directories can contain other directories. Because the INTERACTIVE UNIX System file system is sometimes visualized as a "tree," each new directory created on the system can be viewed as a new branch added to the directory tree. Here is a diagram of a simple directory structure for a computer user named Diane:

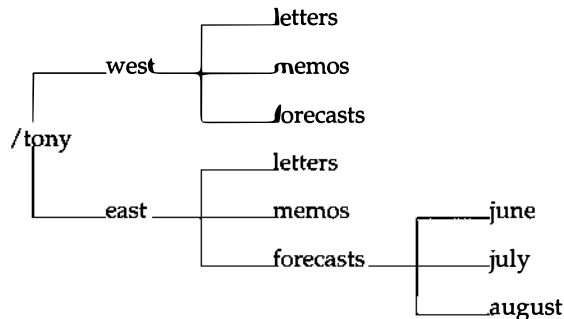


The root directory (/) contains a directory named diane. /diane contains two directories: vendors and orders. These in turn, hold other files and directories. When Diane logs in, she is automatically placed in her home directory, /diane.

Because the INTERACTIVE UNIX System file system is very flexible, the organization of your files and directories can be changed at any time to fit your needs. For example, you might have a subdirectory in your home directory called memos, which contains all the memos you have written. Or, you could have a letters directory, which could contain a number of other directories, one for each of your major projects:



If you deal with different regional offices, you might have a major subdirectory for each region, with each subdirectory containing parallel directories for sales figures, letters, memos, and forecasts:



The only limit on the number of files and directories you can have, or the number of directory levels you can have, is the amount of disk space available to you.

File and Directory Naming Conventions

All programs, text, and data on disks or diskettes reside in files. Each file or directory has a unique name. The INTERACTIVE UNIX System is case sensitive, which means that it distinguishes between upper- and lowercase letters. INTERACTIVE UNIX System file and directory names can be one to fourteen characters long and can include both uppercase and lowercase letters. Although some symbols may be used in INTERACTIVE UNIX System file and directory names, it is safest to limit names to alphanumeric characters (a to z and 0 to 9) and the period (.).

Some symbols have special meanings to the INTERACTIVE UNIX Operating System. Never use an asterisk (*), a question mark (?), a greater than symbol (>), a lesser than symbol (<), an ampersand (&), or a vertical bar (|) in a file or directory name. You will learn more about how these symbols are interpreted by the UNIX System shell later in this guide.

Path Names

When you want to access a directory or file in the file system, you use a *path name*. A path name is a sequence of directory and file names, separated by slashes, that designates the location of a particular file or directory. Since the *full path name* is actually the complete name of the file or directory, it ensures that no two directories (or no two files) on the system have exactly the same full name. To use the filing cabinet analogy again, you could locate a letter addressed to John Smith using its “path name,” by saying it is in the second cabinet drawer, behind the divider marked “Boni Project,” in a folder marked “letters.”

There are two kinds of path names, a full (or *absolute*) path name and a *relative path name*. A full path name *always* begins with a slash (/) and consists of the sequence of directories from the root directory to the file or directory you want to access. A relative path name does *not* begin with a slash. It describes the location of the target file or directory relative to your current location.

A full path name consists of a slash followed by one or more directory names separated by slashes and ending with the name of the target file or directory. The slash at the beginning of the path name tells the system to begin its search for the target from the root directory. For example, the full path name for the file that contains user Tony's eastern region August forecast might be:

```
/tony/east/forecasts/august
```

A relative path name omits all directory names up to and including the current directory from the path name. It tells the system to begin the search for the target file or directory from the current directory. You can access a file in your current working directory by using its relative path name. In this case, the relative path name consists of only the file name. Or, you can use the relative path name to access a file in a subdirectory of your current working directory by specifying the name of the subdirectory followed by a slash and the name of the file. For example, user Diane can display the contents of the `southcity` file in the `orders` subdirectory of her current working directory by using the relative path name with the `cat` command:

```
$ cat orders/southcity
```

It is not necessary to use the full path name to the file `/diane/orders/southcity`.

Path Name Shortcuts: Dot and Dot Dot

The INTERACTIVE UNIX System provides many shortcuts to help make life a little easier and to minimize typing. A relative path name is one type of shortcut. Two other shortcuts are dot (`.`) and dot dot (`..`).

One "dot," when used in place of a directory name, indicates that the command should use the current directory. This shortcut is useful when you are copying or moving files from another directory to your current directory.

For example, you may use the dot (`.`) shortcut to copy a file into your current directory and give it the same name:

```
$ cp /susan/letters/sales.ltr .
```

Two “dots,” when used in place of a directory name, indicate the directory directly *above* your current working directory. The directory above your current directory is also known as the *parent directory*.

You can use `..` to move quickly to a parent directory. For example, if you are currently located in the directory `/tony/east/letters` and want to return to the directory named `east` (the parent directory of your current directory), simply type:

```
$ cd ..  
$ pwd  
/tony/east
```

You can also use a series of “dot dots” to move more than one directory at a time. For example, to move to the root directory from the directory `/tony/east/letters`, type:

```
$ cd ../../..  
$ pwd  
/
```

Common File and Directory Commands

Where Am I (`pwd`)?

If you work in a number of directories on many different projects, you may occasionally forget where you are working. The INTERACTIVE UNIX System command `pwd` (print working directory) will tell you where you are in the directory structure.

COMMAND NAME	<code>pwd</code>
FORMAT	<code>pwd</code>
DESCRIPTION	Print the current working directory. Prints the complete path name of the directory you are currently in.
OPTIONS	None.
ARGUMENTS	None.

Use the `pwd` command to see the full path name of your current directory. If you have just logged in, the system should respond with the name of your home directory:

```
$ pwd
/tony
```

Creating a Directory (`mkdir`)

It is a good idea to organize your files by filing them in subdirectories. Use the `mkdir` command to make new directories.

COMMAND NAME	<code>mkdir</code>
FORMAT	<code>mkdir name(s)</code>
DESCRIPTION	Make a directory with the name given.
OPTIONS	Not presented in this document.
ARGUMENTS	The name(s) of the directory(ies) you want to create.

Use the `mkdir` command to make a new directory called `notes`:

```
$ mkdir notes
```

A `notes` directory is created, but you are *not* automatically moved into it. You remain in the directory from which you issued the command. To verify that the new directory was created, use the `ls` command:

```
$ ls  
notes
```

Remember, press Enter after all commands unless otherwise instructed.

Changing Directories (`cd`)

Use the `cd` command to move from one directory to another.

COMMAND NAME	<code>cd</code>
FORMAT	<code>cd [<i>pathname</i>]</code>
DESCRIPTION	Change current directory. If no argument is given, <code>cd</code> moves you to your home directory.
OPTIONS	None.
ARGUMENTS	The path name (full or relative) of the directory to which you want to move.

For example, to move from your home directory to a subdirectory called `notes`, type:

```
$ cd notes
```

You could use the full path name, but using the relative path name is much easier. To return to your home directory, type `cd`. Using the `cd` command with no arguments always returns you to your home directory. To use the `cd` command with a full path name, type:

```
$ cd /tony/notes
```

What's in This Directory (ls)?

Use the `ls` command to check the contents of a directory. The `ls` command lists all the files and directories in a named directory.

COMMAND NAME	<code>ls</code>
FORMAT	<code>ls [-ltau] [<i>name(s)</i>]</code>
DESCRIPTION	For each directory named, alphabetically list its contents (no options requested) and any other information requested; for each file named, list the requested information.
OPTIONS	<ul style="list-style-type: none"><code>-l</code> List in long format the mode, number of links, owner, group, size, and time of last modification.<code>-t</code> Sort contents by time of last modification (latest first) rather than by name.<code>-a</code> List all entries, including files that begin with a dot (<code>.</code>).<code>-u</code> Use time of last access instead of last modification for sorting (with <code>-t</code> option) or printing on the screen (with <code>-l</code> option).
ARGUMENTS	A file name or a directory name. If no file or directory is named, it gives the current directory.

If your current directory is empty, the `ls` command simply generates a new prompt:

```
$ ls
$
```

The `ls` command accepts a file name, directory name, or path name as an argument. If you use a file name as an argument, `ls` simply lists the name of the file if it exists:

```
$ cd /etc
$ ls profile
profile
```

If you use the `-l` option (as shown in the new command format), `ls` lists more detailed information about the file:

```
$ ls -l profile
-rw-r--r-- 1 bin  other  460 Jun 13  1992 profile
```

Copying a File (`cp`)

The `cp` command is used whenever you need to make a copy of a file. You may want to copy a file if you require several slightly different versions of a letter or if you plan to base a report or presentation on existing material.

COMMAND NAME	<code>cp</code>
FORMAT	<code>cp filename pathname</code>
DESCRIPTION	Copy the named file to the named destination.
OPTIONS	None.
ARGUMENTS	The name of the file you want to copy and its destination.

To create a new file called `employees` that is an exact duplicate of the file `all.emp` in your current directory, use the `cp` command:

```
$ cp all.emp employees
```

The `cp` command may be used to copy a file in someone else's directory to one of your directories, or to copy a file in your directory to another name or location. You must have the read (`r`) permission for a file to copy it from someone else's area. Refer to the `chmod` command in Chapter 9, "Other Frequently Used Commands," for information on file permissions.

To copy a file that resides in another user's directory, you can use a full or relative path name to access the file:

```
$ cp /susan/letters/sales.ltr susan.ltr
$ ls
other.ltr
susan.ltr
```

When you copy a file from another directory, you may use the same file name or you may use a new file name. The only restriction is that you do not already have a file with the same name in your current directory. If you do, you will overwrite the contents of your existing file with the contents of the new file. This happens without a warning from the INTERACTIVE UNIX System.

When you use the copy command to copy the contents of one directory to another directory, the directory you are copying to must already exist.

Removing a File or Directory (`rm`, `rmdir`)

The `rm` command allows you to remove a file or directory you have created.

COMMAND NAME	<code>rm</code> , <code>rmdir</code>
FORMAT	<code>rm [-fri] [filename(s)]</code> <code>rmdir [directory name(s)]</code>
DESCRIPTION	Remove the named file or directory.
OPTIONS	<code>-f</code> Ask no questions about removal. <code>-i</code> Ask before removing each file or directory. <code>-r</code> Examine each directory before removing it.
ARGUMENTS	One or more file or directory names.

To remove a directory, you must first remove all of the files in that directory. Generally, the system will not allow you to remove a directory that contains files.

Use the `ls -a` command to display all files in the `letters` directory, including hidden files (those with file names that begin with a dot—see “Path Name Shortcuts: Dot and Dot Dot” earlier in this chapter):

```
$ ls -a
.
..
.index
lit.ltr
confirm.ltr
```

Use `rm` to remove all the files in the directory except `.` and `..`. These are special file names used only by the system; they cannot be removed. To remove the files:

```
$ rm .index confirm.ltr lit.ltr
```

To remove the `letters` directory, you must `cd` out of it, then use `rmdir` to remove it:

```
$ cd ..  
$ rmdir letters
```


An *editor* is a program that is used to enter and modify text and data. There are two basic types of editors: *line editors* and *full-screen editors*. When using a line editor, you must explicitly specify the line or lines you want to edit. You do not see the changes to your file as they occur. A line editor is useful if you are using a hardcopy terminal, that is, one that does not have a screen. A full-screen editor allows you to see the changes made to a file as you make them. Screen editors are the most popular type of editor today.

Most UNIX Systems have a line editor and one or more full-screen editors. This section outlines three of the most commonly available editors on UNIX Systems; it does not attempt to explain how to use a UNIX System editor.

The ed Line Editor

`ed` is the standard editor provided on every UNIX System. It is a line-oriented editor and was originally developed to run on terminals without screens. `ed` is not the editor of choice for most users since you cannot see changes and additions to a file in context. It uses two command “modes” (insert and command) and, in traditional UNIX System fashion, the editing commands are terse and cryptic. However, it is a very powerful editor for making global changes to a file, and it offers some advantages when you are editing text over slow telephone lines or on a terminal without a terminal definition for the available screen editor. Sometimes it is the only editor available, so it is useful to know a little about `ed`.

The vi Screen Editor

`vi` is one of the most widely used UNIX System screen editors. It was originally developed at the University of California at Berkeley, but it is now distributed as part of USL's UNIX System V Release 3. It is also distributed by many other UNIX System vendors. `vi` is an extremely flexible and powerful editor. It provides many tools and features that programmers find especially useful.

`vi` also uses two modes to perform editing tasks: command and insert. The user must toggle back and forth between the two modes to edit text. All commands, including those that move the cursor keys, must be issued while in command mode. Adding text to a file must be done in insert mode. This means that if you notice an error several lines above your current line, you must change to command mode to move the cursor back up to the error and delete the erroneous characters, return to insert mode to type in the correct text, then change once more to command mode to return to the current work. Some users find that switching modes while editing text can be difficult and time-consuming, while other users have very little trouble using the editor efficiently.

The TEN/PLUS User Interface

TEN/PLUS is a full-screen *user interface* to the UNIX System. It is based on a full-screen editor (the TEN/PLUS editor) developed by INTERACTIVE Systems Corporation to take the mystery out of text editing programs and the UNIX System. It is available on many systems today, including IBM's AIX®.

The TEN/PLUS editor does not rely on modes to perform editing operations; it uses ten basic function keys to perform most editing tasks and is very easy to learn. For a novice user, it is probably much easier to learn TEN/PLUS than the standard UNIX System editors `ed` and `vi`. The TEN/PLUS editor also includes a large number of advanced editing features to make modifying text files easier.

Refer to Chapter 13, "The TEN/PLUS Environment," for more information on how to use the TEN/PLUS editor.

What Is a Shell?

The INTERACTIVE UNIX Operating System consists of two major components, the *kernel* and the *shell*. The kernel is the core of the operating system. It is the part of the *system software* that is responsible for managing the computer's *hardware* and *software* resources and for processing commands. The work that the kernel performs is hidden from most users by the shell.

The shell is the part of the system that interacts with the user. It is also known as the command interpreter and is sometimes referred to as the user interface. It is responsible for passing user commands to the kernel for processing and for returning the results to the terminal screen, into a file, or to another device, such as a printer.

There are currently several versions of the shell available on UNIX Systems. The most popular versions of the shell are known as the *Bourne shell*, the *C shell*, and the *Korn shell*. The Bourne shell was written at Bell Laboratories and released by AT&T (now USL). The C shell was written and released by the University of California at Berkeley. The Korn shell, named after its author, combines features of both shells. The Bourne shell is considered to be the standard UNIX System shell, but the C shell and Korn shell have gained widespread popularity, particularly with programmers. The information in this primer generally applies to all versions of the UNIX System shell.

When you have successfully logged in to your computer, you are automatically using the shell. As you learned earlier, the prompt you see on the screen is also known as the shell prompt. If you do not see a shell prompt and you are sure

that you are logged in to the system, your system may also include a friendly user interface. On some systems, the TEN/PLUS User Interface is included. In this case, the shell is still running, but it is hidden from you in order to provide an even easier method of executing commands and running applications.

Shell Features

Although the shell is an ordinary program that is run by the kernel, it is a very versatile program. Not only does it perform the work of the command interpreter, but it also functions as a programming language. The shell can take commands from special files called *shell scripts* or *shell programs*. A shell script can be a simple collection of shell commands such as those normally executed on the command line, or it can be a program that uses programming logic, variable assignment, and argument processing to perform complex tasks. Shell programming is beyond the scope of this primer, but it is covered in more detail in other documentation resources.

In addition to being a programming language, the shell has a number of distinctive features that permit even the novice user to perform sophisticated operations with a little practice. They include:

- File name shortcuts known as *wildcards*
- The ability to *redirect* the input and output of a command to a file, a printer, or another program
- The ability to use the output of one command as the input to another command, also known as a *pipeline*
- The ability to run two or more programs simultaneously by using *background* processing

Each of these features is described in the following sections.

Using Wildcards

Many INTERACTIVE UNIX System commands require one or more file or directory names as arguments to the command. A wildcard is used to match a character or a string of characters in a file or directory name. It is another type of shortcut. (You have already learned two shortcuts for relative directory names, `.` and `..`.) A wildcard is also referred to as a *pattern matching character*.

Wildcards are handy because they enable you to use a file name or group of file names without having to type the complete name. It is much easier to move all the files from one directory to another using a wildcard than to type out all the file names. Wildcards can also be used when you only remember part of a file name.

There are three wildcard characters:

?

Matches any single character

*

Matches one or more characters

[. . .]

Matches an array of characters

Note – When you create new files, do *not* use a wildcard character as part of the file name. It can be difficult to access or remove a file if there is a wildcard character in its name.

WILDCARD	?
FORMAT	<i>string?</i> <i>?string</i> <i>?string?</i> <i>string??</i>
DESCRIPTION	Substituting a question mark in a file or directory name argument matches one single character for each ? supplied.

A directory called `misc` might contain the following files:

<code>letter1</code>	<code>letter01</code>	<code>letter10</code>	<code>repairs</code>
<code>letter2</code>	<code>letter02</code>	<code>letter.mmb</code>	<code>reporta</code>
<code>letter3</code>	<code>letter03</code>	<code>letter.pc</code>	<code>reportb</code>

To remove the files named `letter1`, `letter2`, and `letter3`, type:

```
$ rm letter?
```

All of the other files will remain, either because they do not start with the string `letter`, or because they have more than one character following the string `letter`.

```
$ ls
letter01
letter02
letter03
letter10
letter.mmb
letter.pc
repairs
reporta
reportb
```

To remove the files named `letter01`, `letter02`, and `letter03`, you cannot use the command `rm letter??` because that would also remove the file `letter10`. Instead, type:

```
$ rm letter0?
```

because it removes only the files with one character following the string `letter0`.

```
$ ls
letter10
letter.mmb
letter.pc
repairs
reporta
reportb
```

WILDCARD	*
FORMAT	<i>string*</i> <i>*string</i> <i>*string*</i> <i>*string*string</i>
DESCRIPTION	Substituting an asterisk in a file or directory name argument tells the computer to match a string of characters (zero or more characters).

To list the names of all files beginning with the letters `re` in the `misc` directory, use the wildcard `*` with the `ls` command. This matches any string of characters beginning with `re`:

```
$ ls re*
repairs
reporta
reportb
```

To list the names of only the files with the character “dot” (`.`) in them:

```
$ ls *.*
letter.mmb
letter.pc
```

Note that while `*.*` refers to all files when using DOS, in UNIX, it refers just to file names with the dot character in them.

Sometimes, it is very useful to use the * wildcard in conjunction with the `rm` command to remove all the files in a directory:

```
$ rm *
```



Warning – This is a very dangerous command! Be *very* sure that you *really* want to remove all of the files in the directory before you use this command. Your files will be permanently lost!

WILDCARD	[...]
FORMAT	<i>string</i> [n-n] [n-n] <i>string</i> [nnn-] <i>string</i>
DESCRIPTION	Substituting a list of characters between square brackets in a file or directory name used as an argument to a command tells the computer to match any characters present in the array.

To list all the letter files in the `misc` directory that end in a number, you could use the [...] wildcard:

```
$ ls letter[0-9]
letter1
letter2
letter3
```

Or, to list only those letter files that end in the numbers 1 and 2, type:

```
$ ls letter[12]
letter1
letter2
```

Redirecting Input and Output

Every command to the computer requires input and produces output. Normally, INTERACTIVE UNIX System commands take their input from the *standard input* and direct it to the *standard output*. The standard input is usually taken from your terminal keyboard. For example, when you type a command and press Enter, you are sending standard input to the kernel. The standard output is usually your terminal screen, although some programs write their standard output to a file or a printer. When you use the `ls` command, for example, the standard input is taken from your terminal keyboard and the standard output is written to the terminal screen.

A very important feature of the INTERACTIVE UNIX System is the ability to direct a command to take its input from a source other than its standard input, and to send its output to a destination other than its standard output. For example, you can direct a command to take its input from a file already stored on the computer and to direct its output to another file or to a device such as a printer. The ability to take input from one place and direct it to another at the time you start a process is called *redirection of input/output (I/O)*.

Two symbols are used to indicate I/O redirection. The “greater than” (>) symbol redirects output, and the “less than” (<) symbol redirects input. Since you will probably redirect the output of a command (>) more frequently than you redirect the input (<), input redirection is not discussed in this primer.

I/O redirection is commonly used to store the output of a command in a file, perhaps to be used with another command or to be incorporated into a file or a message. For example, if you need a listing of all the files in your current directory to use in a report, you could use this command:

```
$ ls > files.list
```


The system automatically creates a new file called `files.list` and writes the output of the `ls` command in that file. You will not see any output on your screen. If you want to verify that the redirection worked, you can use the `cat` command to view the contents of `files.list`:

```
$ cat files.list
files.list
letters
memos
```

To create one large file out of two smaller files, use the `cat` command and redirect the output to a new file:

```
$ cat /etc/passwd /etc/group > users.groups
```

The two files will be written, one after the other, into the file called `users.groups`. The contents of the files `/etc/passwd` and `/etc/group` are not changed, moved, or harmed in any way by this process. The command simply causes a copy of the file's contents to be placed into the named file.

Note – When redirecting output, you must be careful not use an existing file name as your output destination. You can *overwrite* the contents of the existing file if the file's permissions allow it, thereby losing all the text or data in it. (You will learn more about file permissions in the discussion of the `chmod` command in Chapter 9, "Other Frequently Used Commands.")

For example, assume Tony owns a file called `widgets` that is located in his home directory. He wants to put a copy of Barb's eastern `widgets` file into his directory, but he has forgotten that he already has a file named `widgets` in his home directory. If Tony types the command:

```
$ cat /barb/widgets/east.wid > /tony/widgets
```

the contents of his original `widgets` file is deleted and replaced with the contents of Barb's file. The file is still named `widgets`. It is a good idea to check the file names in your destination directory with `ls` before performing an operation like this.

To redirect the output of a command and store the results at the end of a file that already exists *without* overwriting the contents of the file, use two greater than (>>) symbols.

Tony could have *appended* (added to the bottom) the contents of Barb's `east.wid` file onto the end of his `widgets` file with this command:

```
$ cat /barb/widgets/east.wid >> /tony/widgets
```

Using Pipes and Filters

The UNIX System is extremely flexible. It was built on the philosophy that utilities on the system should build on the work of other utilities and be reusable. Two special features are *pipes* and *filters*.

A filter is a program that accepts input from one source, performs the appointed task on the data, and then writes the results to another place without changing the input file in any way. For example, `sort` takes as its

input the file you designate, sorts all of the lines in alphabetical order, and displays it on your terminal. The `sort` command is an example of a simple filter.

COMMAND NAME	<code>sort</code>
FORMAT	<code>sort [-dfn] [-ooutput file] [file name(s)]</code>
DESCRIPTION	Sort the lines of the named file(s) together and write the results to standard output. If no input files are given, it sorts standard input. If no output file is given, it writes to standard output.
OPTIONS	<ul style="list-style-type: none"> <code>-d</code> Use "dictionary" order. Only letters, digits, and blanks are important. <code>-f</code> Ignore case when sorting. <code>-n</code> Sort numeric strings arithmetically.
ARGUMENTS	The name of the file you wish to be the output file, preceded by <code>-o</code> , and the name(s) of the file(s) you wish to sort.

A pipe is a connection between two or more commands. To create a pipeline, you execute a command and pass its output ("pipe" its output) so that it becomes the input for the next command. A pipe is indicated by a vertical bar (`|`). Two filters are often used to create a pipeline. The output of a pipeline can be displayed on the screen or redirected to a file.

The `sort` command is a filter that is frequently used in pipelines. For the following example, assume you have two files: `east.emp` and `west.emp`. The first file (`east.emp`) is a list of employees in the Eastern Regional Office, and the second file (`west.emp`) is a list of all the employees in the Western Region. To create a single file that contains a list of all the employees in the Eastern and Western regions sorted alphabetically, use the following command:

```
$ cat east.emp west.emp | sort -d > all.emp
```

You have taken the output from `cat`, piped it to `sort`, and redirected `sort`'s output into a file called `all.emp`.

To perform the same task but redirect the output of `sort` to a printer instead, use this command line:

```
$ cat east.emp west.emp | sort -d | lpr
```

To learn more about the `sort` command, see the `sort` man page. The `lpr` command is discussed in greater detail in Chapter 9, “Other Frequently Used Commands.”

Background Processing

The INTERACTIVE UNIX System is an *interactive* operating system, which means that you can carry on a dialogue with the computer. Normally, you issue a command and wait for the shell to process your command and return the shell prompt. Sometimes commands can take a long time to complete and do not require any assistance from you. Waiting for the command to finish can be tedious and a waste of time.

To alleviate this, the INTERACTIVE UNIX System allows you to request that a command be run in the background. This means that the system starts the process, displays the process identification number (PID) associated with the process, and then returns the shell prompt. The process continues to run until it is completed, while you continue to use the system for other work. To request background processing, type an ampersand (&) at the end of the command line, immediately before pressing Enter.

Many commands, such as `ls`, complete so quickly that it does not make sense to run them in the background. However, if you are formatting a document or compiling a program, you will find the ability to run a command in the background very useful. Here is an example of a command line where the user requested that it run in the background:

```
$ cat east.emp west.emp | sort -d > all.emp &  
432  
$
```

The shell first gives you an identification number for the process (in this case, 432) and then returns the INTERACTIVE UNIX System prompt. You can now run another command. When your process in the background is finished, the output will be contained in the specified file.

Note that unlike processes being run in the foreground, a background process cannot be temporarily stopped by using Control-d or terminated using Delete. It can be stopped only by logging off or using the `kill` command.

There are literally hundreds of commands available on the INTERACTIVE UNIX System. There are commands to format, extract, and print text in a file; compile, link, archive, and store program code; and report available disk space or commands that are currently running on the system. However, a typical user only needs to use about two dozen commands on a regular basis. We have already described some of the most useful commands in the preceding chapters. This chapter will familiarize you with some other useful commands.

Changing Permissions (chmod)

Every file and directory on an INTERACTIVE UNIX System is associated with a set of *file access permissions*, or *protection modes*, that specify who will have access to it. There are three different types of *permissions*: “read,” “write,” and “execute.”

Read permission

Allows a user to view the contents of the file or copy it to another area. *Read permission* does not allow the user to make changes to the file.

Write permission

Allows a user to make changes to a file. *Write permission* on a directory allows a user to create and delete files in that directory.

Execute permission

Allows a user to execute the file as an INTERACTIVE UNIX System command, if the file is a valid command. *Execute permission* on a

directory allows a user to `cd` into it. If you attempt to `cd` into a directory for which you do not have execute permission, the system will display the message `directoryname:bad directory`.

Each set of permissions applies to one of three categories of users: *owner*, *group*, and *other*. The “owner” of the file owns the file or directory. The “group” often consists of people working on the same project or in the same department and using the same files. Group membership is usually determined by the system administrator. “Other” refers to everyone on the system who is neither the owner nor in the owner’s group.

Table 9-1 shows the default permissions associated with each file and directory.

Table 9-1 Default Permissions

User Category	Directories	Files
User	read, write, execute	read, write
Group	read, execute	read
Other	read, execute	read

Use the `ls -l` command to look at file and directory permissions:

```
total 272
-rw-rw-rw-  1 linda  other    625 Apr  7 13:24 letter01
-r--r--r--  1 linda  other   69979 Apr  2 14:41 letter02
drwxr-x---  1 linda  other     2 Apr  7 13:24 notes
-rw-rw-rw-  1 linda  other   30873 Apr 16 09:02 proj.info
-rwxr-xr--  1 linda  other   69979 Apr  2 14:41 save.memo
```

The access permissions appear on the left of the display.

Now look more closely at the permissions for the file `save.memo`.

File Type	File Access Permissions		
-	rwX	r-x	r--
	User	Group	Other

The first column specifies the file type: a dash (-) indicates a file and a `d` indicates a directory. The first column for the `save.memo` file indicates that it is a file. The next three columns specify the read (r), write (w), and execute (x)

permissions for the owner. In this case, the owner can read, modify, and execute the file. The next three columns specify the permissions set for the owner's group. The permissions indicate that members of the group can read and execute the file. A dash indicates that a permission is denied; therefore, members of the group cannot modify the file. The last three columns specify permissions for others on the system. Others are allowed to read the file but are denied permission to modify or execute it.

COMMAND NAME	chmod
FORMAT	chmod <i>nnn filename(s)</i> chmod <i>nnn directory name(s)</i>
DESCRIPTION	Change mode (permissions) of the named file or directory.
OPTIONS	None.
ARGUMENTS	The octal number or a character representation of the new file access permissions and the name(s) of the file(s) or directory(ies) whose file access permissions are going to be changed.

The `chmod` command takes an argument, which may be in the form of a character representation (*r, w, x*) or a numeric representation (using *octal* numbers) to indicate the permissions in effect.

Each permission is associated with a number: *read=4, write=2, execute=1*, and *no permission=0*. The octal equivalents are obtained by adding together the numbers associated with the basic permissions. Table 9-2 shows the octal equivalents for the basic permissions.

Table 9-2 Octal Equivalents for Basic Permissions

Permission	Octal Representation
-	0
x	1
w	2
wx	3
r	4
rx	5
rw	6
rwX	7

To change the permissions of the `save.memo` file so that members of the group are only allowed to read the file, you can use the `chmod` command with the octal representation:

```
$ chmod 744 save.memo
```

The numbers `744` are the octal equivalent of `rwX, r, r`. This gives the owner read, write, and execute permission in the file but allows only read permission to those in the group and others.

To make the same change using the character representation requires a slightly different approach. Table 9-3 shows the character symbols and their definitions.

Table 9-3 Permission Character Symbols Defined

Permission	Definition
u	Permission applies to user (owner) only
g	Permission applies to group members only
o	Permission applies to others only
a	Permission applies to all (u, g, and o)
+	Add the indicated permissions
-	Remove the indicated permissions
r	Read permission
w	Write permission
x	Execute permission

To change the original permissions on the `save.memo` file so that group members are only allowed to read the file, use this command:

```
$ chmod g-x save.memo
```

The `g` indicates permissions are being changed for members of the group, the minus sign (`-`) indicates removal of a permission, and the `x` indicates execute permission.

To remove read permission for the group and others on the file, use:

```
$ chmod go-r save.memo
```

or:

```
$ chmod 700 save.memo
```

The listing for the `save.memo` file would now look like this:

```
-rwx----- 1 linda other 69979 Apr 2 14:41 save.memo
```

Moving a File or Directory (mv)

On many systems there is a “rename” command and a “move” command. On INTERACTIVE UNIX Systems these two tasks are combined into one command named `mv`. It is used when you want to change the name of a file or directory without making a copy of its contents. It is also used to move the contents of a file or directory from one place in the file system to another.

COMMAND NAME	<code>mv</code>
FORMAT	<code>mv filename pathname</code> <code>mv directoryname pathname</code>
DESCRIPTION	Move the named file or directory to the named position. If a directory name in the current directory is given as the destination for a file, the full path name is not needed.
OPTIONS	None.
ARGUMENTS	The name of the file or directory you wish to move and its destination.

When you move a file from one directory location to another, be sure that you do not already have a file with the same name in the new location. Like the `cp` command, if a file already exists with that name, it is overwritten. The `mv` command may be used to move a file from one directory to another or to rename a file. You must have the correct file access permissions to move a file from another user’s area, that is, you must have read, write, and execute (`rxw`) permission on the directory where the file resides.

To change the name of a file from `all.emp` to `employees`:

```
$ mv all.emp employees
```

The contents of the file remains the same, but it has been renamed `employees`. There is no longer a file named `all.emp`.

```
$ ls
employees
```

To move the `employees` file into an existing subdirectory called `payroll`:

```
$ mv employees payroll
```

Since you are moving a file into a directory, it is not necessary to supply the file name. The system simply moves the file into `payroll`, retaining the same file name. To rename the `employees` file to `dec.pay` and move it into the `payroll` subdirectory at the same time:

```
$ mv employees payroll/dec.pay
```

Changing Owner (`chown`)

If you create a directory, you are the owner of that directory. In some cases, however, you may want to change the ownership of a file or a directory to someone else. Do this by using the `chown` command.

COMAND NAME	<code>chown</code>
FORMAT	<code>chown username filename(s)</code> <code>chown username directoryname(s)</code>
DESCRIPTION	Change ownership of the named file or directory.
OPTIONS	None.
ARGUMENTS	The name of the user whom you want to own the file(s) or directory(ies) and the name(s) of the file(s) or directory(ies) you wish to change the ownership of.

To change the ownership of your file, `local.wid`, to Tony, use the `chown` command:

```
$ chown tony local.wid
```

To change the ownership of all files in the directory to Tony, use the `*` wildcard:

```
$ chown tony *
```

Once you have changed the ownership of a file, your access to that file is limited to what the permissions on that file allow. If you are changing the ownership of a file and want to continue to be able to write in it, be sure the permissions are set accordingly before you use `chown`. Of course, the new owner may change the permissions again.

Checking Spelling (`spell`)

The INTERACTIVE UNIX System includes a utility to check the spelling in a file. The spelling checker is not an interactive command, and it does not provide suggestions for alternative spellings.

COMMAND NAME	<code>spell</code>
FORMAT	<code>spell filename(s)</code>
DESCRIPTION	This command checks the spelling of the input file and lists the words that do not appear in its dictionary on the standard output (the terminal).
OPTIONS	Not presented in this document.
ARGUMENTS	Not presented in this document.

It is a good idea to redirect the output of the `spell` command to another file so that you can refer to it. For example, to check the spelling of the file `notes`, redirect the output of `spell` into the `notes.er` file, then `cat` the `notes.er` file:

```
$ spell notes > notes.er
$ cat notes.er
errir
teh
```

Once you have a list of errors, you can use a text editor to correct the spelling errors. If your text editor allows you to alternate between two files or has a windowing capability, it can be very useful to store the errors in a file that you can refer to while you are making corrections. The system does not recognize the acronyms and other special words that you might use, so `spell` may list some words that you know are not really spelling errors.

Viewing a File (`pg`)

You have already learned about the `cat` command as a way to view a file on the screen. The `pg` command provides another way to view a file on the screen. It is more convenient to use because it displays the text one page at a time to allow you time to read the screen. `pg` displays the next page of text when you press Enter.

COMAND NAME	<code>pg</code>
FORMAT	<code>pg filename(s)</code>
DESCRIPTION	Display the file(s) on the standard output, one page at a time. Use Enter to display the next page. If a file name is not specified, <code>pg</code> reads from the standard input.
OPTIONS	Not presented in this document.
ARGUMENTS	Not presented in this document.

For example, to display the file `letter02`, type:

```
$ pg letter02
```

Printing a File (`lpr`)

After you have created a file, you may want to obtain a printed copy of it. (This printed copy is often referred to as a hard copy.) Use the `lpr` command to print files on your company's printers.

COMMAND NAME	<code>lpr</code>
FORMAT	<code>lpr filename(s)</code>
DESCRIPTION	<code>lpr</code> sends the named file to a printing device.
OPTIONS	Not presented in this document. Most offices have several different printers. See your system administrator for the options you should use to access your printers.
ARGUMENTS	The name of the file(s) you wish to print.

The `lpr` program *queues* each request (puts it in a line for processing) and prints the file as soon as the printer is free. A title page with the system date and time is also printed. `lpr` can be used in a pipeline with other commands if you want to send the output directly to a printer instead of to a file or the terminal screen. For example, to print the file `letter02`, type:

```
$ lpr letter02
```

To get hard copies of two letter files, you can pipe the output of `cat` directly to the printer:

```
$ cat letter01 letter02 | lpr
```

An alternative to the `lpr` command (derived from BSD) is the UNIX System V `lp` command. Both are supplied with the INTERACTIVE UNIX System. See `lpr(1)` and `lp(1)` for details. The use of `lpr` is recommended.

Checking on Processes (`ps`)

To see what processes you are currently running, use the `ps` (report process status) command. This can be very useful when you need to stop a process or to check to see if it has stopped prematurely.

COMMAND NAME	<code>ps</code>
FORMAT	<code>ps [options]</code>
DESCRIPTION	Report the status of all currently active processes on the terminal.
OPTIONS	<code>-f</code> Generate a full listing.
ARGUMENTS	Not presented in this document.

For example, suppose you want to check to see if the `spell` command you started is still running in the background. Type `ps`. Your screen will look similar to this:

PID	TTY	TIME	COMMAND
312	tty01	0:00	ksh
323	tty01	0:00	ksh
325	tty01	0:00	spell
326	tty01	0:01	deroff
317	tty01	0:16	sort
318	tty01	0:00	spellpro
320	tty01	0:00	comm
321	tty01	0:00	tee
327	tty01	0:01	sed
331	tty01	0:00	comm
332	tty01	0:00	tee
333	tty01	0:00	ps

The `spell` process is still running and its process ID number is 325. If you do not see `spell` in the list, then the process either completed running or died before it completed.

Killing a Process (kill)

At times you will want to stop a background process that is currently executing. To do this, use the `kill` command.

COMMAND NAME	<code>kill</code>
FORMAT	<code>kill [-signo] process_ID_number</code>
DESCRIPTION	Terminate the named process.
OPTIONS	<code>-9</code> Send a sure kill signal.
ARGUMENTS	The ID of the process you want to kill.

To stop a process while you are still logged in, type:

```
$ kill id_number
```

where *id_number* is the identification number of the process.

If you do not remember the process ID number, use the `ps` (report process status) command described above to find it. Occasionally `kill` alone is not sufficient to stop a process. If this occurs, type:

```
$ kill -9 id_number
```

If your system includes INTERACTIVE TCP/IP, you can also use the commands that are described in this chapter. These are user-level commands that allow you to copy, execute, or delete files on a remote machine. In some cases, you must supply a *host* (or *rhost*) name as an argument to the command. A table containing host names on your local network is typically kept in the file `/etc/hosts`. This is a system file that contains information, including the host name, for each host on the network. A sample `/etc/hosts` file might look similar to this:

```
127.1      local      localhost
193.16.14.1 market    mktg m # Marketing system
193.16.14.2 train    trng t # Training system
193.16.14.3 accounting acct a # Accounting system running
           # Solaris
193.16.14.4 engineering eng  e # Engineering workstation running
           # Solaris
```

Each entry includes the internet address, the host name, and aliases for the host name. For example, the host name in the first entry is `market`, and the aliases are `mktg` and `m`. You may use either the full host name or one of its aliases wherever command syntax calls for a host name.

To indicate a file or directory on a particular machine, use a colon (:) to separate the host name from the directory or file name. For example, to indicate the `accounts` directory on the host `train`, use the following format:

```
train:/accounts
```

To indicate the `jan.93` file in the `accounts` directory on the `train` machine, separate each part of the path name with a slash (/), as you would in a normal UNIX System path name:

```
train:/accounts/jan.93
```

Remote Login (rlogin)

The `rlogin` command is used to log in to a remote host so that you can work as though you were directly connected to that host.

COMMAND NAME	<code>rlogin</code>
FORMAT	<code>rlogin rhost [-ec] [-l username]</code>
DESCRIPTION	Log in to the named remote host.
OPTIONS	<p><code>-e c</code> Change default escape character, <code>~</code>, to the named character. Do not leave a space between the option and the new escape character.</p> <p><code>-l username</code> Log in to a remote host using a different remote user name. A <code>.rhosts</code> file, containing your name, must exist on the remote host in the login directory of the user whose name you specify. If the <code>.rhosts</code> file doesn't contain your name, you will be prompted for a password. See the following text for more information about <code>.rhosts</code>.</p> <p>Other options not presented in this document. Refer to <code>rlogin(1C)</code>.</p>
ARGUMENTS	The name of a remote host.

A file called `/etc/hosts.equiv` may also be present on each host machine. This file is maintained by your system administrator and contains a list of remote host names. If the `/etc/hosts.equiv` file on the remote host contains the name of your local host and you have a login account on that machine, you can use `rlogin` to log in automatically, that is, without entering a password. (If you have a login account on a remote host but the name of your local host is not specified in the `/etc/hosts.equiv` file on the remote host, you can still log in to the remote host by providing your password.)

For security reasons, your system administrator may not have added your local host's name to the `/etc/hosts.equiv` file on the remote host. You can still log in automatically to the remote host by creating your own private equivalent file on that host. This file is called `.rhosts` and should be kept in your login directory.

Like the `/etc/hosts.equiv` file, the `.rhosts` file should contain the host names of remote machines. This allows you to log in automatically from any host listed in your `.rhosts` file.

You can also use the `.rhosts` file to allow users on other hosts to log in automatically to your account. For example, user `janet`'s `.rhosts` file on the `market` host might look like this:

```
sales
pubs
sales      susan
train      mary
train      john
```

The first two entries indicate that user `janet` can automatically log in to her account on the `market` host from both the `sales` and the `pubs` hosts. The next three entries indicate that user `susan` on the host `sales` and users `mary` and `john` on the host `train` can log in to `janet`'s account on `market`.

To log in automatically to another user's account, specify the `-l` option followed by the login name of the user whose account you wish to access. For example, if user `janet`'s `.rhosts` file on the `market` host contains an entry for user `john` on the `train` host, `john` can log in to `janet`'s account on `market` in this way:

```
train$ rlogin market -l janet
market$
```

Use the Control-d sequence to terminate an `rlogin` session.

The `rcp` and `rsh` commands also make use of the `/etc/hosts.equiv` and `.rhosts` files. Refer to "Remote Copy (rcp)" and "Execute a Shell Command on a Remote Host (rsh)" in this chapter for more information on these commands.

Remote Copy (r`cp`)

The `rcp` command allows you to copy files and directories from one host on the network to another.

COMMAND NAME	<code>rcp</code>
FORMAT	<code>rcp [-p] [host:]file1 [host2:]file2</code> <code>rcp [-r] file directory</code>
DESCRIPTION	Copy the named files and directories from one host to another.
OPTIONS	<code>-p</code> Preserve the modification times and permissions of the original files and directories. <code>-r</code> Copy any subtree (subdirectories and files within a directory) in the named directory. The destination must be a directory.
ARGUMENTS	A source file name and a destination file or directory name.

If you provide only one host name, the local host is used as the default for the unspecified host. For example, user `janet` might copy `file.new` in her current directory on host `market` to the `accounts` directory on the host `train` in this way:

```
market$ rcp file.new train:/accounts
```

Since a source host is not specified, the local host is assumed.

The `rcp` command may also be used to copy files from one remote host to another. User `janet` on `market` could copy the file `/meetings/summary` on the `sales` host to the `accounts` directory on the `train` host by specifying the source host and the destination host:

```
market$ rcp sales:/meetings/summary train:/accounts
```

In order to use the `rcp` command, the `/etc/hosts` file on the remote host must contain an entry for the local host. The `rcp` command also makes use of the `/etc/hosts.equiv` and `.rhosts` files. Refer to “Remote Login (`rlogin`)” in this chapter for more information about these files.

Execute a Shell Command on a Remote Host (`rsh`)

The `rsh` command allows you to connect to the named remote host and execute a shell command. This command is located in `/usr/ucb/`, like most networking commands. It is not the same as `/bin/rsh` (restricted shell).

COMMAND NAME	<code>rsh</code>
FORMAT	<code>rsh rhost [command] [arguments]</code>
DESCRIPTION	Log in to a remote host and execute a shell command.
OPTIONS	<p><code>-l</code> Log in to a remote host using a different user name. A <code>.rhosts</code> file containing your name and the local host name must exist on the remote host in the login directory of the user whose name you specify.</p> <p>Other options not presented in this document. Refer to <i>rsh(1C)</i>.</p>
ARGUMENTS	A host name and a command name.

rsh logs you in to a remote host and executes the named command. If you specify a command, rsh terminates and returns you to your local machine as soon as the command has finished processing. For example, user jack on host market might use the rsh command to get a listing of the files in his login directory on the sales host:

```
market$ rsh sales ls
forecast.95
prospects
plans.new
projects
sample.lit
market$
```

If you do *not* specify a command name, the system logs you in to the remote host and waits for you to issue a command. You will remain logged in to the remote host until you log out from it. To log out of a remote host, use the Control-d sequence:

```
market$ rsh sales
sales$ ls
forecast.95
prospects
plans.new
projects
sample.lit
sales$ Control-d
market$
```

Screen-oriented applications, such as vi, will not run properly with the rsh command. To run vi (or a similar, screen-oriented application), rlogin to the remote host, then run the application.

The rsh command also makes use of the /etc/hosts.equiv and .rhosts files. Refer to “Remote Login (rlogin)” in this chapter for more information about these files. If you are unable to run the rsh command on a remote host, make sure that the /etc/hosts.equiv file on the remote host contains an entry for the local host. See your system administrator if the problem persists. For more information on rsh, refer to *rsh(1C)*.

List Users on Remote Hosts (rwho)

The rwho command lists users logged in to hosts on the local network.

COMMAND NAME	rwho
FORMAT	rwho [-a]
DESCRIPTION	List users on all hosts on the local network. Output includes the host name of the system the user is logged into, the name associated with the user's terminal, the user's login time, and the idle time if greater than 1 minute. Time is specified in the form <i>hours:minutes:seconds</i> .
OPTIONS	-a Include users who have not typed on their terminals for an hour or more. These users are not listed unless this option is specified.
ARGUMENTS	None.

Note – Your system administrator may have disabled the rwho service. If so, the rwho command won't produce any output.

At regular intervals, each machine on a local network broadcasts information about users who are logged in. This information is stored in system files. The rwho command displays information from these system files on the screen. A typical rwho listing might look similar to this:

alan	market:tty10	Jul	25	11:08:01
bob	sales:tty02	Jul	25	09:23:22
diane	train:tty09	Jul	25	08:15
jane	sales:tty05	Jul	25	08:22:34
john	train:tty01	Jul	25	10:02
ken	engineering:tty02	Jul	25	15:31:33
mary	train:console	Jul	25	13:20:02
susan	sales:tty06	Jul	25	08:45:12
tom	engineering:tty04	Jul	25	10:30:14

If no broadcast information has been received from a remote host for 11 minutes, *rwho* assumes that the host is down and does not include users on that host in its output.

File Transfer Program (*ftp*)

The *ftp* command allows you to transfer files between your local host and a remote host, including a non-UNIX System host, as long as that host is connected to the network and uses TCP/IP protocols.

COMMAND NAME	<i>ftp</i>
FORMAT	<i>ftp</i> [<i>-vni</i>] [<i>rhost</i>]
DESCRIPTION	Transfer files between a local host and a remote site.
OPTIONS	<i>-v</i> Display all responses from the remote server and report on data transfer statistics. <i>-n</i> Turn off auto-login feature. <i>-i</i> Turn off interactive prompting during multiple transfers. Other options not presented in this document. Refer to <i>ftp(1C)</i> .
ARGUMENTS	A host name.

You may invoke the *ftp* command with or without a remote host name. If you do not specify a remote host when you invoke *ftp*, you must use the *open* subcommand in order to connect to a remote host. To determine which remote hosts are available to you and what the correct host names are, refer to the */etc/hosts* file on your system or see your system administrator.

When you type the *ftp* command, *ftp* enters its command mode and displays the *ftp>* prompt. In this mode, you can issue any of the *ftp* subcommands (see the partial list of subcommands below). These commands behave like standard UNIX System commands and use standard UNIX System syntax. At

the `ftp>` prompt, type `?` to view a complete listing of the `ftp` subcommands. To view a one-line description of a particular subcommand, type `help` followed by the subcommand name.

Once you make a connection to a remote host and log in, you can use any of the `ftp` subcommands to exchange files or obtain directory or status information. You can end an `ftp` session with the `quit` or `bye` subcommands, either of which will end the session and exit `ftp`.

Some commonly used `ftp` subcommands include:

`cd remote-directory`

Change the working directory on the remote host to *remote-directory*.

`delete remote-file`

Delete the *remote-file* on the remote host.

`get`

Retrieve the *remote-file* and store it on the local machine.

`ls [remote-directory] [local-file]`

Print an abbreviated directory listing of *remote-directory*. If *local-file* is specified, place the listing in that file. If *local-file* is not specified, display the listing on the terminal.

`mkdir directory-name`

Make a directory on the remote host.

`put local-file [remote-file]`

Store a local file on the remote machine.

`quit`

Terminate the `ftp` session and exit `ftp`.

`remotehelp [command-name]`

Request help from the remote `ftp` server.

A simple ftp session to list the files on the remote host train from the local host market might look similar to this:

```
market$ ftp train
Connected to train.
220 train FTP server (Version 5.60) ready.
Name (train:janet):
331 Password required for janet.
Password:
230 User janet logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 36
-rw-r--r--  1 janet  other      1625 Apr  6 20:10 course.new
-rw-r--r--  1 janet  other      2897 Apr  6 20:11 hands.on
-rw-r--r--  1 janet  other      2643 Apr  6 20:10 literature
-rw-r--r--  1 janet  other      7111 Apr  6 20:10 products
-rw-r--r--  1 janet  other      2801 Apr  6 20:11 sales
226 Transfer complete.
ftp> delete hands.on
250 DELE command successful.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 30
-rw-r--r--  1 janet  other      1625 Apr  6 20:10 course.new
-rw-r--r--  1 janet  other      2643 Apr  6 20:10 literature
-rw-r--r--  1 janet  other      7111 Apr  6 20:10 products
-rw-r--r--  1 janet  other      2801 Apr  6 20:11 sales
226 Transfer complete.
ftp> quit
221 Goodbye.
market$
```

Refer to *ftp(1C)* for a complete list of ftp subcommands.

Remote Login Using Telnet Protocol (telnet)

The `telnet` utility is an interactive program that allows you to log in to a remote host as though you were directly connected to that host. Unlike `rlogin`, `telnet` allows you to connect to any remote host, including a non-UNIX System host, as long as that host is connected to the network and uses TCP/IP protocols. This can be very useful, for example, if you want to log in to a non-UNIX host from a host that is running the UNIX System.

COMMAND NAME	<code>telnet</code>
FORMAT	<code>telnet [rhost] [port]</code>
DESCRIPTION	Perform remote login using the Telnet protocol.
OPTIONS	None.
ARGUMENTS	A remote host name and a port number.

To use `telnet`, you must know the name of the remote host you wish to access. To determine which remote hosts are available to you and what the correct host names are, refer to the `/etc/hosts` file on your system or see your system administrator. In addition to specifying a host name, you may specify a *port* number for the remote host. A port is a particular physical connection to a host and is identified by a port number.

If you specify a remote host name without specifying a port number, `telnet` uses a default port number. Once you are connected to the remote host, you enter `telnet`'s input mode. In this mode you can type text and commands as you normally would if you were logged directly in to the remote system. In

input mode, you can also execute the `telnet` subcommands (see the partial listing of `telnet` subcommands below). To execute a subcommand from input mode, you must first type `Control-]` (Control-Right Bracket).

```
market$ telnet sales
Trying...
Connected to sales.
Escape character is '^]'.

System V UNIX (sales)

login: janet
Password:
INTERACTIVE UNIX System V/386 Release 3.2, Version 4.1
sales
Copyright (C) 1994 Sun Microsystems, Inc.
Copyright (C) 1988 AT&T
Copyright (C) 1988 Microsoft Corp.
All Rights Reserved
Login last used: Wed Apr 6 15:10:10 1994
sales$ ls
forecast.95
prospects
plans.new
projects
sample.lit
sales$ Control-]
telnet> status
Connected to market
Escape character is Control-].
sales$
```

Alternatively, you can use the `telnet` command without specifying an *rhost* or *port* number. If you run `telnet` without an argument, you will enter `telnet`'s command mode, indicated by the `telnet>` prompt. In this mode, `telnet` accepts and executes any of the standard `telnet` subcommands (see the partial listing of `telnet` subcommands below).

Some commonly used telnet subcommands include:

? [*command*]

help [*command*]

List a summary of help information for telnet subcommands. If a command is specified, help information for just that command will be listed.

escape [*new character*]

Set the escape character to the named *new character*. Prompts for escape sequence if not specified.

open *host* [*port*]

Open a connection to the named host.

quit

Close an open telnet session and exit telnet.

status

Show connect/disconnect status.

To use a subcommand from command mode, simply type the subcommand at the telnet> prompt. For example, to find out whether you are connected to the remote host, use the status subcommand:

```
market$ telnet
telnet> status
No connections.
Escape character is '^'.
telnet>
```

Note that in this example, no connection has been made because the telnet command was used without a remote host name. A connection is only made when a remote host name is specified as an argument to telnet (telnet *rhost*) or as an argument to the open subcommand (telnet> open *rhost*).

The following example illustrates how to use the `open` subcommand to connect to a remote host from the `telnet` command mode:

```
market$ telnet
telnet> open sales
Trying...
Connected to sales.
Escape character is '^]'.

System V UNIX (sales)

login: janet
Password:
INTERACTIVE UNIX System V/386 Release 3.2, Version 4.1
sales
Copyright (C) 1994 Sun Microsystems, Inc.
Copyright (C) 1988 AT&T
Copyright (C) 1988 Microsoft Corp.
All Rights Reserved
Login last used: Wed Apr 6 15:33:56 1994
sales$
```

To quit `telnet` from input mode, use the Control-d sequence. To quit `telnet` from command mode, use the `quit` subcommand. For more information on `telnet` and a complete listing of `telnet` subcommands, refer to *telnet(1C)*.

The INTERACTIVE UNIX Operating System contains a very useful feature, the *virtual terminal*, which allows you to log in several times simultaneously on a console terminal. When you switch from one virtual terminal to another, your screen clears and the processes you started in another login session continue to run in the background even while out of view. In one session you can be editing a file, while in others you can be running different programs.

Virtual terminals can be enabled only on a console terminal. By default in the INTERACTIVE UNIX Operating System, you can have one session on a console terminal plus two other virtual terminals. This number can be changed by the system administrator; up to seven virtual terminals can be enabled, for a total of eight possible login sessions.

The virtual terminals that are enabled on your system are always ready for use. On most enhanced PC keyboards, it is necessary to press the Alt key to perform a Sys-Req function.

1. To switch to the first virtual terminal, press the Alt and Sys-Req keys simultaneously, then release them both and press F1.
2. When the new prompt appears, you may log in with your usual login ID and password:

```
VT1 Login:  
Password:
```

3. To switch to the second virtual terminal, again press Alt and Sys-Req simultaneously, then release them and press F2. Similarly, pressing Alt and Sys-Req simultaneously, followed by F3, F4, F5, F6, or F7 invokes virtual terminals 3 through 7, respectively. If the corresponding terminal is not enabled on your system, the terminal will beep instead.
4. To return to the system console, press Alt and Sys-Req simultaneously, then release them and press either F8 or R.
5. You can also cycle forward through all the currently enabled virtual terminals by repeating the following sequence: press Alt and Sys-Req simultaneously, then release them and press F10. You can cycle backward through all of the currently enabled virtual terminals by repeating the following sequence: press Alt and Sys-Req simultaneously, then release them and press F9.
6. Each virtual terminal behaves as if it were an entirely separate terminal sitting on your desk. Therefore, in order to be completely logged out of the system, you must log out of each virtual terminal that you have logged in to.

The INTERACTIVE UNIX Operating System provides two mechanisms for accessing MS-DOS (DOS) files from the INTERACTIVE UNIX Operating System: the DOS File System Support (DOS-FSS) and the Dossette File Exchange Utility (`dossette`).

DOS-FSS allows a DOS file system to function as an integral part of the INTERACTIVE UNIX System partition. This facility allows multiple users to simultaneously access the DOS file system. DOS files need not be copied into the INTERACTIVE UNIX System partition but can be manipulated directly, using path names that are similar to those used on the INTERACTIVE UNIX System. And INTERACTIVE UNIX System commands and utilities can be used with DOS files, just as they would be used with any INTERACTIVE UNIX System file. DOS-FSS is recommended for high-volume situations that require access to and manipulation of many DOS files, using INTERACTIVE UNIX System commands and utilities. DOS-FSS can be used with both fixed disk partitions and diskettes.

`dossette` provides a specific set of commands for manipulating DOS-format file systems and for moving files between DOS and INTERACTIVE UNIX System file systems. `dossette` allows DOS files to be copied into the INTERACTIVE UNIX System partition where they can be edited or otherwise manipulated, then copied back into the DOS file system. Format conversions are performed automatically. `dossette` also includes commands for creating DOS directories, deleting DOS files, and printing DOS ASCII files on the terminal screen. `dossette` is easily invoked and provides a convenient

method for accessing DOS files from the UNIX System in low-volume situations that require a limited amount of file manipulation. `dossette` can be used with both fixed disk partitions and diskettes.

Using DOS-FSS

Under DOS-FSS, a DOS file system must be *mounted* on an INTERACTIVE UNIX System directory before it can be accessed; and a DOS file system must be *unmounted* in order to ensure that all data is written to disk. Since superuser privileges are required to mount and unmount a file system, these procedures are usually performed by your system administrator.



Caution – Do not remove a DOS file system diskette from the diskette drive until it has been unmounted, or you may lose data.

See your system administrator to ensure that your DOS file system is mounted and unmounted correctly. If you are the system administrator, refer to the *INTERACTIVE UNIX System Maintenance Guide*.

DOS File Names

The syntax used for DOS path names is different from that used for INTERACTIVE UNIX System path names. The INTERACTIVE UNIX System uses a series of directory names and (optionally) a file name, each separated by a slash (/). DOS uses a drive name (*device specifier*) followed by a series of directory names and (optionally) a file name, each separated by a backslash (\). (A *device specifier* usually consists of an alphabetic character followed by a colon. For example, the device specifier `a:` usually indicates the first diskette drive, and the device specifier `b:` usually indicates the second diskette drive.)

Your system administrator will mount your DOS file system under an INTERACTIVE UNIX System directory name, such as `/dos`. See your system administrator for the appropriate path name to your DOS file system.

Once the DOS file system is mounted under DOS-FSS, DOS files can be accessed in the same way that INTERACTIVE UNIX System files are accessed. DOS files now function as part of the INTERACTIVE UNIX System file system, so path names need not specify a device name. Instead, they follow the conventions for INTERACTIVE UNIX System path names, that is, they are

made up of directory and file names, each separated by a slash. The backslash character is not valid. For example, to look at the DOS file `autoexec.bat` in the `/dos` directory, you could use this command:

```
$ cat /dos/autoexec.bat
```

Note that the path name does not specify a device name and that the components of the path name are separated by slashes.

Under DOS, all file names are mapped to uppercase, regardless of whether they are typed in uppercase or lowercase. DOS-FSS provides this same mapping on input so that users accustomed to this feature can continue to specify DOS files using lowercase. Similarly, DOS file names are displayed in lowercase on output.

DOS-FSS also expands wildcard characters (`*` and `?`). For example, the DOS file `EXAMPLE.TXT` residing in the current directory could be found using any of the following:

```
$ ls e*
$ ls e*.*
$ ls exam???.*
$ cat e*.txt
```

Refer to Chapter 8, “Using the Shell,” for more information about wildcards.

File Conversion Utilities

Text files are stored differently under the DOS and UNIX Systems. The INTERACTIVE UNIX System uses the “new-line” character to specify the end of a line; DOS uses a “carriage-return” followed by a “line-feed” to specify the end of a line. Because of this, DOS files viewed through an INTERACTIVE UNIX System editor display control characters at the end of each line. DOS-FSS provides three utilities to convert DOS files to UNIX System format and UNIX System files to DOS format:

`dtou`

Converts DOS files to UNIX System format.

`utod`

Converts UNIX System files to DOS format.

`lef`

Determines the current format of the specified text file and converts it to the other format.

All three utilities take zero, one, or two arguments. If no argument is specified, the utility takes input from standard input (usually your keyboard) and sends output to standard output (usually your display screen). If one argument is specified, it is assumed to be the input file, and the output is sent to standard output. If two arguments are specified, the first is taken to be the input file and the second is taken to be the output file. The following examples illustrate the command format of the conversion utilities.

This command converts the DOS text file `example.txt` to a UNIX System text file and displays it on the standard output:

```
$ dtou /dos/example.txt
```

This command converts the UNIX System text file `/tmp/doc.txt` to a DOS text file in the `/dos` directory:

```
$ utod /tmp/doc.txt /dos/doc.txt
```

This command uses the `lef` utility to determine the format of the `doc.txt` file, converts it from the current format to the other format, and places the new file in the `/tmp` directory:

```
$ lef /dos/doc.txt /tmp/doc.txt
```

Permissions

The INTERACTIVE UNIX System is a *multi-user* environment that makes use of such concepts as owners of and permissions for files and directories. (Refer to Chapter 9, "Other Frequently Used Commands," for more information about file owners and file permissions.) DOS is a *single-user* system and has no

equivalent concepts. In order to support multi-user access to a mounted DOS file system, DOS-FSS attempts to apply these UNIX System concepts to the DOS file system.

If you cannot access a file or directory on your DOS file system, the permissions may have been set incorrectly. See your system administrator. If you are the system administrator, refer to the *INTERACTIVE UNIX System Maintenance Guide* for more information about permissions.

Using dossette

Device Specifiers

Like DOS, dossette recognizes certain device specifiers. For example, the specifier for the first diskette drive is `a:`, the specifier for the second diskette drive is `b:`, and the specifier for the DOS partition on the first fixed disk is `c:`. If you attempt to access a diskette or fixed disk partition that is not in DOS format while using dossette, an error message will be displayed. When dossette is initially invoked, the current device is always set to `a:`, but it can be changed in the usual DOS fashion by entering a different device specifier, followed by Enter.

DOS File Names

The format for a DOS file name includes an optional device specifier, an optional path name, and the name of the file. If the device specifier is omitted, the current device is assumed. If no path name is specified, the current directory on the specified device is assumed. The components of a path name are each separated by a slash (/). (The backslash character is not valid in dossette.) For example, the file name `temp` indicates the file `temp` in the current directory on the current DOS device; the file name `b:/util/temp` indicates the file named `temp` in the directory `util` in a DOS file system on the second diskette drive.

Interactive Commands

When the dossette command is typed at the prompt, dossette is invoked in interactive mode. In this mode, commands that are similar to DOS commands are read from standard input, and each line is executed as it is read.

`dossette` prompts the user for input, as necessary. The program mimics DOS in issuing a prompt at the beginning of each line to indicate the current device, for example, `A>`, `B>`, `C>`. To invoke `dossette` interactively, type `dossette` at the shell prompt:

```
$ dossette
A>
```

`dossette` is now ready to receive any of the following commands:

`dir`

Lists DOS directory information.

`type`

Prints a DOS ASCII file on the terminal screen.

`copy`

Copies a DOS file to a new DOS file name. You may also copy a DOS file to a DOS directory.

`rename`

Changes the name of a DOS file.

`erase, del`

Deletes a DOS file.

`mkdir, md`

Creates a DOS directory.

`rmdir, rd`

Removes a DOS directory.

`format`

Physically formats a DOS diskette. This command does not format a fixed disk partition.

`get`

Copies a DOS file or files to a UNIX System file system and changes the file or files to UNIX System format.

`put`

Copies a UNIX System file or files to a DOS file system and changes the file or files to DOS format.

- `cd`
Changes or displays the DOS working directory.
- `chloc`
Changes the INTERACTIVE UNIX System working directory.
- `a:`
Changes the current DOS device to drive A, which corresponds to the first diskette drive.
- `b:`
Changes the current DOS device to drive B, which corresponds to the second diskette drive.
- `c: . . . q:`
Changes the current DOS device to the fixed disk partition corresponding to the device specifiers `c:` through `q:`.
- `help`
Displays a list of available `dossette` commands on the screen.
- `q`
Exits `dossette`.

The following examples illustrate how to use the `dossette` commands interactively.

The `get` command allows you to copy a DOS file into the INTERACTIVE UNIX System partition and converts the DOS control characters to UNIX System format:

```
A> get b:/util/tmp /src/unix.tmp
```

In this example, the DOS `tmp` file, located in the `util` directory on the second diskette drive, is copied to the UNIX System file, `unix.tmp`, in the UNIX System `src` directory. The file is converted to UNIX System format. Once the file is converted in this way, it can be edited using a UNIX System editor.

The `put` command allows you to copy a UNIX System file back to the DOS partition or diskette and converts the file to DOS format:

```
A> put /src/unix.tmp b:/util/dos.new
```

In this example, the `unix.tmp` file, located in the `src` directory, is copied to the DOS file, `dos.new`, in the DOS `util` directory on the second diskette drive. The file is converted to DOS format.

To exit `dossette`, type `q` at the prompt.

Batch Mode

The `dossette` command group is also available for use in *batch* (noninteractive) *mode* so that you can use the `dossette` commands in a shell script. In this mode, the `dossette` command line consists of the invoking command and its arguments, if any. The `dossette` batch commands are prefixed by `dos`; they operate in the same way as the interactive `dossette` commands described previously. The `dossette` commands available in batch mode are: `dosdir`, `dostype`, `doscopy`, `dosrename`, `dosmkdir`, `dosrmdir`, `dosformat`, `dosget`, and `dosput`.

This example illustrates how to use the `dosget` batch command. Note that the command and its arguments are typed at the shell prompt:

```
$ dosget b:/util/tmp /src/unix.tmp
```

In this example, the DOS `tmp` file, located in the `util` directory on the second diskette drive, is copied to the UNIX System file, `unix.tmp`, in the INTERACTIVE UNIX System `src` directory. The file is converted to UNIX System format.

The TEN/PLUS environment makes the INTERACTIVE UNIX Operating System easy to learn and easy to use. This chapter introduces you to the ten basic *functions* of the TEN/PLUS environment, which involve:

- Creating and storing documents, such as letters and reports
- Retrieving documents
- Revising documents
- Setting up new filing systems
- Performing complex tasks, such as moving information between documents
- Printing documents

Note – As you read about the TEN/PLUS environment, refer to “Summary of TEN/PLUS Functions” at the end of this chapter. It lists the function keys and how they are used in the TEN/PLUS environment. Note, too, that TEN/PLUS functions appear in all capital letters in this chapter.

Getting Started

Your system may be installed so that a TEN/PLUS screen will automatically appear when you log in. If the system displays a prompt such as \$ or % when you log in, access the TEN/PLUS environment by typing e \$HOME and pressing Enter. If you are still unable to enter the TEN/PLUS environment, ask your system administrator to help you.

Your Home Directory

The first screen you see when you log in to the system is usually your *home directory*. Your home directory can contain subdirectories (folders) that hold related files (documents) and other subdirectories. For example, you can have a subdirectory called `letters` that contains all the outgoing letters you've written. The `letters` subdirectory can also contain a number of other subdirectories, one for each of your major projects. For example, there could be a `quik.sell` subdirectory under `letters` that contains all the letters related to the QuikSell project. Here is a typical home directory listing:

```

TEN/PLUS FILE MANAGER
Move the cursor to an item below and ZOOM-IN to see it.

```

File	Description
Report	First Report on the QuikSell Project
policies	Company Policies File
letters	Outgoing Letters Directory

```

/usr/larry          INSERT   Line   1 ( 3)

```

File

If you are a new user, your home directory may not show any *files* because you have not yet created any. Take a moment to study the display shown above, paying particular attention to the labels identifying specific elements on the screen. (Some screens may differ in minor ways.)

Cursor

The *cursor* is a pointer to where the next character will appear. As you type, each character appears at the cursor position, and the cursor moves one space to the right. Entering text is similar to typing with a

typewriter. However, in most cases in the TEN/PLUS system, when you reach the right margin, the cursor and the word being typed move automatically to the next line. This feature is called *word wrap*. You can, of course, end a line before you reach the right margin by pressing ENTER, which moves the cursor to the beginning of the next line.

The File field

This *field* contains the names of the files and directories stored in the displayed directory. A file usually contains a single document, such as a report or a memorandum. A directory is also a file, but it is a special type of file that can contain other files.

You should limit file and directory names to 10 characters. File names can contain any characters other than], [, *, ?, /, and space, and they must not start with a hyphen (-) or a plus (+).

The Description field

This field allows you to supply a short description of the file or directory.

Full Path Name

The *full path name* completely describes the location of the file in the system. Slashes are used to separate directory names and file names. The full path name is displayed at the bottom of the screen.

Using ZOOM-IN and ZOOM-OUT

You can use the TEN/PLUS functions ZOOM-IN and ZOOM-OUT to move around within a directory structure. ZOOM-IN moves you to a lower level in the directory structure and ZOOM-OUT moves you to a higher level. To move from a directory to a file or directory in the next lower level, position the cursor on the line on which the file or directory is listed and ZOOM-IN. To move from a file or directory to the directory in the next higher level, ZOOM-OUT. It does not matter where the cursor is positioned in the file or directory when you ZOOM-OUT.

Using the Cursor-Positioning Functions

The *cursor-positioning functions* move the cursor on the screen. In addition to ENTER (discussed in a previous subsection), the most frequently used cursor-positioning functions are ↓, ↑, →, and ←; they move the cursor down, up,

right, and left, respectively. You can also use BACKSPACE to move the cursor to the left, but, unlike ←, BACKSPACE erases characters as it moves. BACKSPACE is used to correct typing errors.

TAB moves the cursor to the next tab stop on the right, while -TAB moves the cursor to the previous tab stop on the left. On most terminals, the cursor-positioning functions repeat automatically. Holding down the key(s) for any of these functions will result in continuous cursor motion until the keys are released. You can use the cursor-positioning functions to place the cursor anywhere on the screen, including on any previously typed character, as well as on the borders of any field. (If you place the cursor on such a border and attempt to type there, you'll get an error indication, usually a "beep.")

Creating Documents

You can create documents (also called files) in any of your directories, including your home directory. Suppose you want to create a file called `status`, with the description `Current Sales Status`, and that you want this file to be in your home directory. To create the file, move the cursor to the first

blank line on your home directory screen. Type the file name, `status`, in the File field. Use TAB or one of the other cursor-positioning functions to move the cursor to the Description field, then type `Current Sales Status`:

```

                                TEN/PLUS FILE MANAGER
                                Move the cursor to an item below and ZOOM-IN to see it.

                                File                               Description
-----
status                          Current Sales Status_
-----
/usr/larry                        INSERT   Line   1 ( 1)
```


Now ZOOM-IN. The TEN/PLUS system determines that a file named `status` does not exist in your home directory, and shows you a menu with four options:

```

TEN/PLUS FILE MANAGER
Move the cursor to an item below and ZOOM-IN to see it.

File                                     Description
-----
status | Current Sales Status
^
| You are attempting to create file "/usr/larry/status"
| Move the cursor to the type of file you want and touch EXECUTE.
| Touch CANCEL to do nothing, HELP for help.
|
| Create an ASCII file (without history)
| Create a structured file (with history)
| Create a directory
| Re-enter the file name
|
-----
/usr/larry                                INSERT   Line   1 ( 1)

```

The options shown allow you to create an *ASCII file*, a *structured file*, or a directory, or to re-enter the file name in case you misspelled it. Usually, you will create ASCII files, which are ordinary text files. The examples in this primer use only directories and ASCII files. Structured files allow you to keep a record of different versions of a document, and to recreate any prior version. Structured files are particularly useful when you need to look at past versions of a document, such as a legal contract or a business plan. Many TEN/PLUS applications, such as the TEN/PLUS Mail System, use structured files.

With the cursor on the option Create an ASCII file (without history), use EXECUTE. After a brief pause, a blank window appears:

```
l   t   t   t   t   t   t   t   t   r  
~  
/usr/larry/status          INSERT   Line   1 ( 0)
```

You have just created a file with the short name `status`. The line at the top of the screen is a “ruler” showing the positions of the left and right margins (l and r) and of the tab stops (t). (Instructions for changing margins and tab stops appear later in this chapter.)

Type some text into your new file. When you complete the last line on the screen, the text will *scroll* forward (up), bringing in additional blank lines at the bottom of the screen. You can also scroll forward by using either `+PAGE` or `+LINE` to move beyond the area displayed on the screen. `+PAGE` scrolls text forward by one screen, and `+LINE` scrolls text forward by approximately one-third of a screen. Similarly, `-PAGE` and `-LINE` scroll text backward (down).

Use `ZOOM-OUT` to move from the `status` file back to your home directory. It does not matter where the cursor is positioned in the file when you `ZOOM-OUT`.

Using *HELP* and *CANCEL*

HELP provides details about menus and *popup boxes*. Use *HELP* whenever you are uncertain about what to do. Either a *popup box* with additional instructions or a menu of options will appear on your screen.

CANCEL allows you to remove menus and *popup boxes* from your screen. For example, if you decide that you do not want to create a new file, use *CANCEL* when the file creation menu appears. This causes the menu to be removed from the screen.

To see how *HELP* and *CANCEL* work, *ZOOM-IN* to the *status* file, and then *ZOOM-IN* again. The message *Cannot zoom in any further* appears in a *popup box*. Use *HELP*. The error message is explained in more detail in another *popup box*. Use *CANCEL* to remove the *popup boxes*.

Creating Directories

You can create directories using the same procedure that you used to create new files. Create a directory called *practice* in your home directory. Type *practice* in the *File* field of your home directory, then *TAB* to the *Description* field and type *Practice Files Directory*:

```

          TEN/PLUS FILE MANAGER
    Move the cursor to an item below and ZOOM-IN to see it.

```

File	Description
status	Current Sales Status
practice	Practice Files Directory_

```

/usr/larry                                INSERT   Line   2 ( 2)

```

Now ZOOM-IN. The file creation menu appears. Select the option to create a directory. After a brief pause, a blank directory screen appears:

```
TEN/PLUS FILE MANAGER
Move the cursor to an item below and ZOOM-IN to see it.

File                Description
┌──────────┬───────────────────────────────────────────────────────────────────────────────────┐
│           │                                                                                   │
│           │                                                                                   │
│           │                                                                                   │
│           │                                                                                   │
│           │                                                                                   │
│           │                                                                                   │
│           │                                                                                   │
│           │                                                                                   │
│           │                                                                                   │
│           │                                                                                   │
└──────────┴───────────────────────────────────────────────────────────────────────────────────┘

/usr/larry/practice          INSERT      Line 1 ( 0)
```

Now you can type in the names and descriptions of your practice files, and use ZOOM-IN to create them, just as you created the status file in your home directory.

Editing Files and Directories

The TEN/PLUS environment offers a range of editing tools based on the use of six TEN/PLUS functions:

- PICK-UP
- PUT-DOWN
- PICK-COPY
- PUT-COPY
- INSERT
- FORMAT

Using PICK-UP and PUT-DOWN

PICK-UP is used with PUT-DOWN to move text. To see how this is done, create the file priorities in the practice directory. Enter this text into the file:

```
l   t   t   t   t   t   t   t   t   t   r
Priority list of tasks to be accomplished before Account Group meeting.

    Obtain cost figures for three inventory profiles
    Review copy for ad campaign
    Memo to George about comments for ad campaign
    Write outline for presentation
    **Reminder -- follow up on getting projection charts from
    Art Department_

/usr/larry/practice/priorities      INSERT   Line   11 ( 11)
```

To move the line `Write outline for presentation` to the top of the list, position the cursor on that line and use `PICK-UP`. The entire line is picked up and disappears from the screen:

```
l   t   t   t   t   t   t   t   t   t   r
-----
Priority list of tasks to be accomplished before Account Group meeting.

Obtain cost figures for three inventory profiles
Review copy for ad campaign
Memo to George about comments for ad campaign
*Reminder -- follow up on getting projection charts from
  Art Department

/usr/larry/practice/priorities      INSERT   Line   9 ( 10)
```

Note that PICK-UP picks up the entire line, regardless of where the cursor is positioned, and moves the subsequent lines up. To move the line to the top of the list, position the cursor on the line Obtain cost figures for three inventory profiles, then use PUT-DOWN:

```

71      t      t      t      t      t      t      t      t      t      r

```

```

Priority list of tasks to be accomplished before Account Group meeting.

Write outline for presentation
Obtain cost figures for three inventory profiles
Review copy for ad campaign
Memo to George about comments for ad campaign
**Reminder -- follow up on getting projection charts from
Art Department

```

```

/usr/larry/practice/priorities          INSERT   Line   6 ( 11)

```

You can also transfer information from one file to another by using PICK-UP and PUT-DOWN in conjunction with ZOOM-IN and ZOOM-OUT. First, ZOOM-IN to a file and PICK-UP the line you want. Then, using ZOOM-OUT and ZOOM-IN as appropriate, access the directory containing the file into which the line is to be placed. ZOOM-IN to this file and use PUT-DOWN to insert the line where you want it.

You have just seen how to PICK-UP and PUT-DOWN one line of text at a time. You can use PICK-UP several times in succession, followed by PUT-DOWN the same number of times, to move several lines of text at one time. PICK-UP can also be used to delete text—simply PICK-UP that text and do not put it down.

Using PICK-COPY and PUT-COPY

PICK-COPY and PUT-COPY are similar in function to PICK-UP and PUT-DOWN. However, PICK-COPY leaves the original line where it was, and only picks up a copy. PUT-COPY allows you to put down several copies of the last line picked up by using either PICK-UP or PICK-COPY.

Create a file named copiers in your practice directory, then enter this text:

```
l   t   t   t   t   t   t   t   t   t   r
```

Evaluation chart for buying new copier				
BRAND	COST	SPEED	QUALITY	SERVICE
Hybrid-II	\$2,300	100 per min.	B+	A-
Comments: Has enlargement/reduction features, LED displays. Paper refill easy, toner refill somewhat clumsy.				

```
/usr/larry/practice/copiers
```

```
INSERT Line 10 ( 10)
```


In this example, you typed in information about one brand of copier, Hybrid-II. To use the same headings (BRAND, COST, SPEED, QUALITY, SERVICE) to begin a new section for another brand, move the cursor to the heading line and use PICK-COPY. Then move the cursor to where you want to start typing your evaluation of the second brand, and PUT-DOWN the copy:

```

i   t   t   t   t   t   t   t   t   t   r
Evaluation chart for buying new copier
BRAND      COST      SPEED      QUALITY     SERVICE
Hybrid-II  $2,300      100 per min.  B+          A-

           Comments: Has enlargement/reduction features, LED displays.
                   Paper refill easy, toner refill somewhat clumsy.

BRAND      COST      SPEED      QUALITY     SERVICE
/usr/larry/practice/copiers          INSERT   Line   14 ( 14)

```

In this example, you could have used PUT-COPY instead of PUT-DOWN to continue putting copies of the heading at the beginning of each new section, without having to use PICK-COPY each time. PUT-COPY inserts a copy of the same line until you pick up another line by using either PICK-UP or PICK-COPY.

PICK-UP, PUT-DOWN, PICK-COPY, and PUT-COPY can also be used to move, copy, and delete entire files and directories. For example, you can delete a file or directory from a directory by placing the cursor in either field on the line where the file is listed and using PICK-UP.



Warning – Deleting a directory deletes all files and directories in that directory!

You can transfer a file from one directory to another, perhaps from your home directory to another directory containing several related files. This procedure is similar to that for transferring data between files. Use ZOOM-IN and/or ZOOM-OUT to access the directory containing the file you want to transfer. Position the cursor on the line where the file is listed, and PICK-UP the file; move to the destination directory and PUT-DOWN the file. You can also use this procedure to move directories.

PICK-COPY and PUT-COPY can be used to copy—rather than move—files and directories. For example, you might want to maintain a master version of a form letter that can be copied and completed as needed. Position the cursor on the line where the file is listed and use PICK-COPY. Next, move to the destination directory and use PUT-DOWN. If you copy a file into its original directory, a popup box appears, asking you to give the copy a new name. This is because two files in the same directory cannot have the same name. Type a new file name (for example, `form.1tr2`) in the popup box and use EXECUTE.

Using INSERT and FORMAT

Two other basic TEN/PLUS functions are INSERT and FORMAT. They are used to insert space for new text, and to format text between current margins.

Create the file memo in your practice directory, then enter the text as shown:

l t t t t t t t t r

Ellen:

I wanted to solicit your thoughts about the possibility of Lisa attending the Sales Conference next week. I know that she's working on getting the display materials ready for the TransCorp presentation next month, but if you could spare her, I think it would be a valuable experience for her internship and an asset to us if we hire her upon graduation. One of the problems with a lot of MBAs when they get out of school is that they don't have a good feel for the personal side of business. I think it would be good for Lisa to get some of this exposure. What do you think? Will the TransCorp schedule permit?_

/usr/larry/practice/memo

INSERT Line 14 (13)

The body of this example can be split into two paragraphs to make it more readable. Move the cursor to the O in the word One on the fifth line of the paragraph, then use the spacebar until the phrase One of the has been moved to the next line:

l t t t t t t t t r

Ellen:

I wanted to solicit your thoughts about the possibility of Lisa attending the Sales Conference next week. I know that she's working on getting the display materials ready for the TransCorp presentation next month, but if you could spare her, I think it would be a valuable experience for her internship and an asset to us if we hire her upon graduation.
One of the problems with a lot of MBAs when they get out of school is that they don't have a good feel for the personal side of business. I think it would be good for Lisa to get some of this exposure. What do you think? Will the TransCorp schedule permit?

/usr/larry/practice/memo

INSERT Line 10 (14)

Note – If touching the spacebar does not move the text to the right, read “Alternating Between Insert and Overwrite Modes” later in this chapter.

Use a blank line to separate paragraphs. With the cursor on the line One of the, use INSERT. A blank line is inserted:

```

l   t   t   t   t   t   t   t   t   t   r
-----
Ellen:

I wanted to solicit your thoughts about the possibility of Lisa attending
the Sales Conference next week. I know that she's working on getting the
display materials ready for the TransCorp presentation next month, but if
you could spare her, I think it would be a valuable experience for her
internship and an asset to us if we hire her upon graduation.

One of the
problems with a lot of MBAs when they get out of school is that they don't
have a good feel for the personal side of business. I think it would be
good for Lisa to get some of this exposure. What do you think? Will the
TransCorp schedule permit?

/usr/larry/practice/memo          INSERT   Line   10 ( 15)

```

You can use INSERT to insert as many blank lines as you want. For example, you can use INSERT to insert space for new paragraphs, or to reserve space for diagrams on a printed copy of a file.

In the above example, the second paragraph needs to be reformatted because it begins with a short line. Position the cursor on the short line and use FORMAT. FORMAT reformats text to fit within the current margins:

```
l   t   t   t   t   t   t   t   t   t   r
|-----|
| Ellen:                                     |
|                                           |
| I wanted to solicit your thoughts about the possibility of Lisa attending |
| the Sales Conference next week. I know that she's working on getting the |
| display materials ready for the TransCorp presentation next month, but if |
| you could spare her, I think it would be a valuable experience for her   |
| internship and an asset to us if we hire her upon graduation.           |
|                                           |
| One of the problems with a lot of MBAs when they get out of school is that |
| they don't have a good feel for the personal side of business. I think it |
| would be good for Lisa to get some of this exposure. What do you think?  |
| Will the TransCorp schedule permit?                                       |
|                                           |
|-----|
/usr/larry/practice/memo                INSERT   Line   11 ( 14)
```

Note – FORMAT reformats text from the current cursor position to the next blank line. Be sure to leave one or more blank lines between paragraphs as you type. Otherwise, FORMAT will run all of your paragraphs together!

Using Menus

The TEN/PLUS environment includes two functions, MENU and LOCAL-MENU, that simplify the way you perform more complex tasks.

Using MENU

MENU offers a menu, called New Task Menu, that provides a number of general-purpose options. Your New Task Menu displays the same options regardless of which file or directory you are looking at. Here is a typical example of a New Task Menu:

```

TEN/PLUS FILE MANAGER
Move the cursor to an item below and ZOOM-IN to see it.

File                                     Description
-----
priorities                               Priorities for the Account Group Meeting
copier                                   Copier evaluations
memo                                     Letter to Ellen about Lisa

New Task Menu
Move the cursor to an item and touch EXECUTE.
Touch CANCEL to do nothing, HELP for help.
Show home directory
Display the current date and time
Read or send mail
Show your profiles directory
Edit your editor profile
Housekeep
Display history of current file

/usr/larry/practice                      INSERT   Line   3 ( 3)

```

The options on your New Task Menu may differ somewhat from those displayed above. You can easily add or remove options from your New Task Menu. The New Task Menu displayed above contains these options:

- Show home directory
Returns you to your home directory, regardless of where you are in the directory structure.

Display the current date and time

Displays the current date and time in a popup box. (Use CANCEL to remove the box.)

Read or send mail

Displays your mailbox if the TEN/PLUS Mail System is available on your computer.

Show your profiles directory

Displays your profiles directory so you can view or edit your TEN/PLUS profiles.

Edit your editor profile

Displays your editor profile, which is used to customize your editing environment.

Housekeep

Removes all versions of files except the current version.

Note – You must use Housekeep periodically (for example, once a day) to prevent various files from growing too large and wasting storage space.

Display history of current file

Displays a list of all versions of the current file.

To select an option from this or any other menu, position the cursor on the desired option and use EXECUTE. As a shortcut you can use one of the TEN/PLUS functions (1) through (8), depending on the number that corresponds to the line on which your choice is listed. For example, to select Show your profiles directory from the default New Task Menu, use (4), since this is the fourth option on the menu. Use CANCEL to remove a menu from the display.

Using LOCAL-MENU

LOCAL-MENU is similar to MENU except that it displays a menu of options that apply specifically to the type of information or to the application you are using at the moment. Here is a sample local menu for an optional TEN/PLUS application, the TEN/PLUS Mail System:

Subject: Forwarded: First thoughts on the QuikSell account
 To: larry
 Cc: brian
 Bcc: Janet Brown
 Date: 18 Apr 1992 0957-PDT
 From: Janet Brown

I have extracted the juiciest parts here ... let me know what you think.
 ^

Electronic Mail Move the cursor to an item and touch EXECUTE. Touch CANCEL to do nothing, HELP for help.	they want to accomplish with soon with a couple of t down here my best notes of my own on things we ls. Also, I have included orrespondence from the a better idea of what has u have looked at all of this, questions you may have about roach that will have a very if our proposals are very
(1) Mail this message (2) Reply to this message (3) Forward this message (4) Delete this message (5) Restore deleted message (6) File this message in another mailbox (7) Show in-box and add new mail	

Size: 150 Lines Sent by: janet at RALEIGH Status:
 /usr/larry/mhs/1 INSERT Line 6 (150)

If you use LOCAL-MENU when looking at a directory, this menu will appear:

TEN/PLUS FILE MANAGER
Move the cursor to an item below and ZOOM-IN to see it.

File	Description
priorities	Priorities for the Account Group Meeting
copiers	Copier evaluations
memo	Letter to Ellen about Lisa
^	
Move the cursor to desired action and touch EXECUTE. To do nothing, touch CANCEL. For help, touch HELP.	
- (1)	Display "visible" files
(2)	Display all files
---	Return to normal directory display
(4)	Show details about files
(5)	Show more details about this file
---	Show more details about this file

`/usr/larry/practice` INSERT Line 3 (3)

This local menu is seldom used in simple applications.

More About Editing

In addition to the basic TEN/PLUS functions, several other editing functions are also quite useful. They are used to change margins and tab stops, to provide alternate methods for modifying or editing existing text, to find a specific word or phrase in a file, and to move text between files. The sections below explain how to use these functions, as well as how to print your documents.

Changing Margins and Tabs

Up to now, you have used the *default* margins and tabs that are set automatically. You can change these margins and tabs at any time.

To change the left margin, position the cursor where you want the new left margin to be and use MARGIN. The `l` on the ruler on the top line moves to the new left margin. To type an indented paragraph, for example, move the left margin in, type the paragraph, and then move the left margin back to its original position.

To change the right margin, position the cursor where you want the right margin to be and use ENTER then MARGIN. The `r` on the ruler on the top line moves to the new right margin.

You may have occasion to type a line or lines of text that extend beyond the right border of the screen. To do this, you will need to change the right margin to a column greater than 77 (the default column position for the right margin). You can use RIGHT to bring into view a portion of the file (approximately one-third of the width of the screen) that extends beyond the right border. Using RIGHT again will bring another such portion into view. Bring as much of the file into view as necessary to reset the right margin. The maximum setting of the right margin is column 200.

Similarly, you can use LEFT to bring into view a portion of the file (if any) that extend beyond the left border.

You can use BEGIN-LINE to move the cursor to the leftmost character of the current line, and END-LINE to move the cursor one position to the right of the rightmost character of the current line.

To set a new tab stop, position the cursor at the desired column and use SET-TAB. To remove a tab stop, use TAB to position the cursor on that stop, then use ENTER followed by SET-TAB.

Alternating Between Insert and Overwrite Modes

Until now, you typed text while in insert mode. When insert mode is in effect, new text is inserted at the cursor position, and existing text is moved to the right or word-wrapped to the next line. The word INSERT on the bottom line of the screen indicates that insert mode is in effect.

On the other hand, new text replaces existing text as you type when overwrite mode is in effect. The word OVERWRITE replaces the word INSERT on the bottom line of the screen.

You can switch between insert and overwrite modes with INSERT-MODE. If you are in insert mode, INSERT-MODE places you in overwrite mode; conversely, if you are in overwrite mode, INSERT-MODE places you in insert mode.

Note – INSERT-MODE is independent of INSERT. INSERT-MODE is used to switch between insert and overwrite modes, while INSERT is used to insert blank lines in a file or directory.

Using +SEARCH, -SEARCH, and BREAK

On occasion, you may want to quickly locate a word or phrase in a file. To search for a word or phrase, starting at the current cursor position and continuing to the end of the file, use ENTER, type the word or phrase you want to find, then use +SEARCH. The system searches for an exact character match. For example, a search for the word sales will find sales, but not Sales, because the small s and the capital S are different characters. If there is no matching word or phrase, a message will appear in a popup box. If the search is successful, the cursor will be positioned on the matching word or phrase. You can then search for the next occurrence of the same word or phrase by simply using +SEARCH again.

You can also search backward through the file, starting at the current cursor position and continuing to the beginning of the file. Use -SEARCH instead of +SEARCH to search backward.

You can interrupt a search operation by using **BREAK**. For example, if you discover after using either **+SEARCH** or **-SEARCH** that you made an error while typing the word or phrase to be found, you can use **BREAK** to stop the search operation.

Using USE

As you have seen, you can move through the directory structure to a specific file by using **ZOOM-OUT** and **ZOOM-IN** as appropriate. Because you are able to see the directory and file names at the various directory levels, you do not need to remember the full name of a file in order to access it.

You can, however, use another method to access a specific file directly, if you know its exact name. Use **ENTER**, type the name of the desired file in the popup box, and use **USE**. **USE** brings the desired file onto the screen. At this point, you have established what is known as an alternate file. Now every time you use **USE** the system will alternately display the original file and the alternate file.

To directly access another file, or to establish a new alternate file, simply use **ENTER**, type the name of the new alternate file, and use **USE**. Once you have established an alternate file, you can move text between the original file and the alternate file by using **PICK-UP** or **PICK-COPY** in one file, then **USE**, and then **PUT-DOWN** or **PUT-COPY** in the other.

Printing Documents

To print a document, **ZOOM-IN** until its contents are visible, then use **PRINT**. A menu appears, listing the print options available on your system.

To print a document on your default printer, position the cursor at the option **Print on default printer**, and use **EXECUTE**.

Summary of TEN/PLUS Functions

This section summarizes the ten basic functions, as well as a number of additional functions, that are available with the TEN/PLUS system. Many of them can be modified or enhanced when used with other functions.

The Ten Basic Functions of the TEN/PLUS System

The keys used on a PC keyboard to perform these functions appear in parentheses.

MENU

Displays the New Task Menu. (F1)

LOCAL-MENU

Displays a menu of options specific to the current application or to the type of information you are using. (F2)

FORMAT

Formats the current paragraph within the current margins. (F3)

INSERT

Inserts a blank line at the current cursor position. (F4)

PICK-UP

Removes the current line and saves it until it is PUT-DOWN. (F5)

PUT-DOWN

Inserts, at the current cursor position, the last line picked up. (F6)

PICK-COPY

Picks up a copy of the current line and moves the cursor down one line. (F7)

PUT-COPY

Inserts, at the current cursor position, a copy of the last line picked up. (F8)

ZOOM-IN

Displays a more detailed level of information.

ZOOM-OUT

Displays a less detailed level of information.

Additional TEN/PLUS Functions

↑, ↓, →, ←

Move the cursor up, down, right, or left.

BACKSPACE

Corrects typing errors by erasing characters.

BEGIN-LINE

Moves the cursor to the leftmost character of the current line.

BOX-MARK

Marks lines or rectangular blocks of text for copying, inserting, or deleting. Can be used with cursor-positioning and other TEN/PLUS functions, such as PICK-UP, PICK-COPY, INSERT, and DELETE.

BREAK

Stops +SEARCH and -SEARCH.

CANCEL

Removes an error message or popup box from the display.

CENTER

Centers the current line between the current margins.

DELETE

Deletes the current line. The deleted line can be restored with RESTORE.

DELETE-CHARACTER

Deletes the character at the current cursor position.

END-LINE

Moves the cursor one space to the right of the rightmost character of the current line.

ENTER

Enhances or modifies many TEN/PLUS functions, such as PICK-UP, PICK-COPY, INSERT, and DELETE. For example, ENTER 5 PICK-UP picks up five successive lines of text.

EXECUTE

Invokes options that appear in menus or “help” popup boxes. Cursor-positioning functions are used to place the cursor at the option to be EXECUTEd.

EXIT

Exits from the TEN/PLUS environment, saving all changes made to files during this editing session, or since the last SAVE.

FUNCTIONS

Displays a menu showing which of the ten basic TEN/PLUS functions are active and allows you to select one.

GO-TO

Displays the beginning of the file and positions the cursor on the first line. If the cursor is already on the first line of the file, GO-TO displays the end of the file and positions the cursor on the last line of the file. ENTER GO-TO always displays the end of the file and positions the cursor on the last line of the file. ENTER line-number GO-TO moves the cursor to that line of the file. For example, ENTER 47 GO-TO displays the portion of the file that contains line 47 and positions the cursor on that line.

HELP

Displays information relevant to the current situation.

HOME

Moves the cursor to the upper left corner of the display.

INSERT-MODE

Alternates between insert and overwrite modes.

LEFT

Brings into view a portion (approximately one-third of the width of the screen) of the file that extends beyond the left border.

+LINE

Scrolls the text forward (up) approximately one-third of the screen at a time. ENTER *n* +LINE scrolls the text forward *n* lines. For example, ENTER 3 +LINE scrolls the text forward three lines.

-LINE

Scrolls the text backward (down) approximately one-third of the screen at a time. ENTER *n* -LINE scrolls the text backward *n* lines. For example, ENTER 4 -LINE scrolls the text backward four lines.

MARGIN

Sets the left margin at the current cursor position. ENTER MARGIN sets the right margin.

+PAGE

Scrolls the text forward (up) a full screen at a time. ENTER *n* +PAGE scrolls the text forward *n* full screens. For example, ENTER 2 +PAGE scrolls the text forward two full screens.

-PAGE

Scrolls the text backward (down) a full screen at a time. ENTER *n* -PAGE scrolls the text backward *n* full screens. For example, ENTER 3 -PAGE scrolls the text backward three full screens.

PRINT

Displays a menu of print options.

REFRESH

Redraws the screen display.

RESTORE

Restores, at the current cursor position, the most recently DELETED text.

ENTER

Moves the cursor to the beginning of the next line.

RIGHT

Brings into view a portion (approximately one-third of the width of the screen) of the file that extends beyond the right border.

SAVE

Saves all changes made to files during this editing session, or since the last SAVE.

+SEARCH

Searches forward through the file for a specified word or phrase. For example, ENTER Sam +SEARCH searches for the next occurrence of the word Sam; subsequent occurrences of Sam can be found by using +SEARCH alone.

-SEARCH

Searches backward through the file for a specified word or phrase. Like +SEARCH, -SEARCH can be used alone to find subsequent occurrences of the word or phrase.

SET-TAB

Sets a tab stop at the current cursor position. ENTER SET-TAB removes the tab stop at the current cursor position.

TAB

Moves the cursor to the next tab position on the right.

-TAB

Moves the cursor to the previous tab position on the left.

TEXT-MARK

Marks text for copying, inserting, or deleting. Can be used with cursor-positioning and other TEN/PLUS functions, such as PICK-UP, PICK-COPY, INSERT, and DELETE to manipulate multiline sentences and nonrectangular text regions.

USE

Used with ENTER to directly access another file. Also used to switch back and forth between two files.

This chapter explains the *internationalization* features of the INTERACTIVE UNIX Operating System and describes how to use it on computer systems outside the United States (U.S.), where there are differences in local language, customs, and standards. This document focuses on usability and is restricted to languages that have an alphabet of fewer than one hundred letters. Korean, Japanese, Chinese, and other languages with thousands of different letters are not supported by the standard INTERACTIVE UNIX Operating System. In certain countries, SunSoft, Inc.'s distributors sell a special version of the product to accommodate these special markets. Contact your sales representative for more information.

To find out how to set up a user to use the system in an international environment, refer to the *INTERACTIVE UNIX System Maintenance Guide*.

Background

Computers and their method of operation have generally been associated with American English. Until recently, computer users and programmers accepted the fact that operating and programming a computer had to be in English.

Internationalization is the art of making a computer, a computer system, or a computer program (often called an application) function in a non-U.S. environment. The word "internationalization" itself illustrates that the different behavior a computer system must support not only depends on the use of a different language, but also on the country of origin, even if the language is the same. Spelling may be different; for example, in American

English the word is spelled internationalization, while in England the spelling is internationalisation. To avoid the spelling problem, the acronym *I18N* is becoming common (whether in the U.S. or England, internationalization begins with the letter I, ends with N, and has 18 letters in between).

Some people confuse internationalization with making language-specific (for example, French) versions of applications. But *I18N* does not refer to the *translation* of software, but rather to its usability and translatability. An internationalized application or computer system is one that can be tuned to different environments. The term *localization* (and its acronym *L10N*) is used to describe the adaptation of computer programs to a single language and/or country, which, if mismanaged, can be as costly as making a separate version for each language.

Entering Data

The UNIX System is an interactive, multi-user, time-sharing operating system, which means that several computer users interact with the computer at the same time, usually by typing on a keyboard. This input, as well as the result of the computations done by the application used, is in turn displayed on the computer screen as output.

The device used to interact with the computer is usually either a self-contained unit with a keyboard and a screen that is connected to a serial port of the computer (a *terminal*) or a directly connected keyboard and a monitor attached to the computer's video card, usually referred to as the *console*.

Input consists of keystrokes that typically represent letters and other symbols, which are pictured on the keys of the keyboard. A computer, however, speaks no particular language and has no notion of what a letter is. Instead, a letter is stored in a computer (either in its memory or in a file on the fixed disk) as a number. Unless every computer system uses the same number to store a certain letter, much confusion is created when attempting to transfer data from one type of machine to another. For that reason, conventions and standards for storing characters into a computer have been created. For more information about this, refer to "Storing Data in the Computer" in this chapter.

Most keyboards today have 101 or 102 keys. These keys can be divided into three groups:

- The main section of the keyboard
- The numeric keypad
- The function keys

The main section of the keyboard contains keys used to type regular letters and punctuation characters, such as period (.) and semicolon (;). The layout of this section of the keyboard differs from country to country.

The numeric keypad is a section of the keyboard designed for easy and fast access to all the numeric characters (0–9) and symbols indicating operators, such as plus (+) and asterisk (*). The numeric keypad is often compared to the keys on a calculator. This set of keys can be used in two modes. In the first, they generate the numerals and symbols pictured on the keycaps; in the second, they act as special function keys and cursor movement keys. The mode in effect is indicated by the Numlock light and can be changed by using the Numlock key. When the Numlock light is on, the keys generate the numerals and symbols on the keycaps.

The layout of the function key section of the keyboard depends on the manufacturer, but today most computer keyboards are relatively standard. They usually contain 10 or 12 function keys on the top row of the keyboard, labeled F1 to F10 or F12. These keys generate sequences of characters, such as Escape o p, often called *escape sequences*. (Escape is the code generated by the Escape key.)

Applications can take advantage of these keys by determining the actual escape sequence generated by a function key through the `termcap` or `terminfo` interfaces. These interfaces allow the development of terminal-independent applications.

The layout of both the numeric keypad section and the function key section of the keyboard is the same regardless of the country in which a specific keyboard is used.

U.S. Personal Computer Keyboard Layout

The main section of a keyboard designed for use in the United States contains keys for all letters of the English alphabet, all digits, and the most commonly used punctuation characters and special symbols. Some of these symbols, the

slash (/), for example, are especially important when using the INTERACTIVE UNIX Operating System. In addition, a few special modifier keys are present. The Shift key, when pressed simultaneously with a letter key, generates an uppercase character instead of a lowercase character, or alternate symbols instead of the numbers and symbols on the top row.

The Caps-Lock key exchanges uppercase and lowercase. In other words, when this key is pressed, it changes the state of the keyboard so that all characters subsequently typed are automatically uppercase and only appear in lowercase when pressed together with the Shift key. The Caps-Lock light indicates the status of the keyboard.

The spacebar generates a space character to put one or more spaces between words. Other special keys are Tab, Alt, Enter, and Backspace. For technical details about these keys, refer to the manual entry *keyboard(7)*.

The layout of the keyboard is basically the same as on most typewriters. The layout is often referred to as *QWERTY*, after the order of the first five letters on the top row of keys containing letters. By using the same layout on all typewriters and terminal keyboards, computer users can type in text at a very high speed, regardless of the equipment they are using.

Although one might expect that the layout was chosen to give the easiest access to the most frequently used characters, this is not the case. The *QWERTY* keyboard layout was originally designed to be slow enough so that mechanical typesetting machine operators would not be able to type fast enough to jam their machines.

Generating Characters Not Present on a U.S. Keyboard

Although non-English characters like the German ä or the French ê are not present on a keyboard designed for use in American English, most of these characters can be generated. This allows anyone to write French letters on American systems, for example. There are three ways to generate characters for which there are no keycaps (explicit symbols on the keyboard):

- Deadkeys
- Compose sequences
- The decimal representation of the character

Deadkeys

The *deadkey* was invented by typewriter manufacturers. For example, imagine you need the French character ê. A French typewriter does not have a key for this character, but it has keys for both e and ^ . When the key ^ is pressed, a circumflex is printed but the typewriter carriage does not move. When the e key is then pressed, the letter “e” is printed on the same spot as the circumflex and an ê is formed. This technique works very similarly on a terminal. The only difference is that when ^ is pressed, nothing happens until e is pressed, after which the character ê appears on the screen.

A utility that can be used to assign deadkeys, `ttymap`, is supplied with the INTERACTIVE UNIX Operating System. This utility is used to do everything discussed in this section. To define ^ as a deadkey and try the other examples listed below, type the command:

```
$ ttymap /usr/lib/keyboard/usa.map
```

Now when you press ^ nothing appears on the screen. When an e is typed next, the letter ê appears. To use the ^ character alone, press ^ first and then the spacebar. If a sequence of two characters is typed that does not make sense at all, no character is sent to the application that is currently being used, and the terminal beeps to indicate that an erroneous combination was typed.

Compose Sequences

Although assigning deadkeys supports more characters than the ones printed on the keyboard, it has its disadvantages. As illustrated above, it is annoying when you need the specific character alone that has been assigned as a deadkey. Instead of one keystroke, two keystrokes are needed to access that character. If too many keys act as deadkeys, the system becomes difficult to use.

Fortunately, another method, often referred to as *compose sequences*, exists. A special key or sequence of keys is used to put the keyboard into a special mode. We will call the key or key sequence the Compose key and the special mode the Compose mode. The default Compose key sequence for the INTERACTIVE UNIX Operating System is Control-Shift-F1. (Many DOS users will be familiar with it.) When in Compose mode, the system expects two more characters to be typed by the user before a character is generated. Press

Control-Shift-F1 followed by `n ~` to produce the Spanish ñ (the ñ in mañana) on the screen. If you press the Compose key sequence followed by pressing ! (exclamation mark) twice, an inverted exclamation sign appears on the screen.

Both the value of the Compose key and the list of Compose key sequences and the characters they generate can be specified in a file that is then processed by the `ttymap` command. Refer to `ttymap(1)` for more details.

Some terminals, for example, the DEC® VT220™, have a dedicated Compose key on the keyboard, and the characters are generated by the terminal hardware.

Decimal Representation

A third method of generating characters is using their *decimal representation*. As explained in “Storing Data in the Computer” later in this chapter, every character corresponds to a unique number. Up to 256 different characters can be used (although some terminals only support 128). When the Compose key is used, followed by three digits, the character that is internally represented by the three-digit number (in decimal) is generated. This feature is also derived from the DOS system. Press the Compose key sequence, followed by 065, and an A appears on the screen. 65 is the decimal value used by computers to store the uppercase letter A. Press the Compose key sequence followed by 136 and the letter ê appears.

If you type:

```
$ ttymap -d
```

all deadkeys and compose sequences are disabled.

Smiling Faces

Those familiar with personal computers and certain DOS applications may have seen interesting images the size of a character, such as smiling faces or musical notes. When Control characters are used (characters generated by pressing Control and a letter key simultaneously), normally nothing is displayed on the screen. However, when the Escape key is pressed before pressing Control and a letter, an image appears on the screen (note this only works on the console). For example, Escape-Control-a produces a smiling face.

European Personal Computer Keyboard Layouts

In Europe, computers are sold with either U.S. keyboards (to be used with very technical, engineering-style applications, usually in English) or keyboards designed for the local country. These keyboards differ from U.S. keyboards in the following ways:

- Keyboard layout
- 102 rather than 101 keys

The extra key is usually located between the Shift key and the leftmost bottom row key (Z on a U.S. keyboard). In most countries, this key has the angle bracket characters, < and >, printed on it. In addition, the backslash key (\) on U.S. keyboards, typically the rightmost or second rightmost key in the top row of the main keyboard section, is usually moved to the left of the Enter key in the third row. The layout usually is the same as the one found on typewriters used in these countries. They are often named after the order of the first five keys on the second row of keys; keyboards used in France are called *AZERTY* keyboards, and keyboards used in Germany are called *QWERTZ* keyboards.

Most Western European languages have an alphabet that contains only a few more letters than English (usually no more than 12 extra letters). For example, French uses all the letters used in English, as well as a number of accented characters, such as é, è, and á. Some of the characters, such as the ê used in previous examples, are accessed using a deadkey; most of the others are printed on a keycap.

The keys that are used for symbols, such as the square bracket ([]) and curly brace ({} on U.S. keyboards, have local language accented characters printed on them rather than the American characters. Although not often used in text, these symbols are certainly important in the context of the UNIX Operating System, especially when the system is used for C programming. Having sacrificed these symbols to support the local language, there must be an alternative way of obtaining them. The solution provided by most keyboard manufacturers is to print three symbols on the top row keys. In addition to the digits and symbols, such as plus (+) and minus (-), the braces and brackets are printed either in the right bottom corner or on the front of the keycap. To generate these symbols, press the key simultaneously with the right Alt key.

Note – When using most applications on the INTERACTIVE UNIX Operating System, no distinction is made between the left and the right Alt key, but in certain applications, such as those based on X11, a distinction *is* made.

In the INTERACTIVE UNIX Operating System, `ttymap` input files are provided for all major European keyboards. When the system is properly configured by the system administrator, keyboards function correctly without user intervention, even before logging in to the system.

Keyboards to be used in France and Switzerland require special attention. On French keyboards, the Shift key must be used to access the digits printed on the top row. A Swiss keyboard can be used in two modes. It has keys with four characters printed on it (the same two characters are printed twice, but in opposite order). In German Swiss mode, German characters like ö are accessed by pressing a key, French ones like á by using the Shift key as well. In French Swiss mode, it works the opposite way.

Russian or Greek Keyboards

Certain languages, such as Greek or Russian, use completely different alphabets. Although they may look similar, the Russian and Greek alphabets do differ. What they have in common is that they consist of a reasonably small set of letters (31 for Russian and 24 for Greek) and that, although some of the letters also exist in English, all of these letters are considered separate from the English set. A personal computer keyboard that supports these languages is designed differently than the ones discussed in the previous section.

The remainder of this section discusses a keyboard designed to support both U.S. English and Russian (use with Greek is practically the same). A U.S. English/Russian keyboard (other variants, such as German/Russian keyboards, exist) is physically identical to U.S. English keyboards. The only difference is that in addition to the English letters, the Russian letters are also pictured on the keycaps, usually in a different color. Using `ttymap`, the keyboard is mapped to generate Russian characters when a key is pressed. A special key, called a toggle key, can be used within an application to switch between Russian and English. The default sequence for toggling between languages is Control-Shift-F2.

This feature of the INTERACTIVE UNIX System terminal subsystem and the `tty` utility has been especially designed to support languages such as Greek and Russian. The same toggle key can be used with European keyboards to temporarily cause deadkeys to no longer act like deadkeys, for example. A French programmer might decide to use the toggle key when he switches between a C source code file and a French text file, for example.

Keyboard Layouts on 7-bit Terminals

The keyboards described so far are keyboards that are attached to devices capable of supporting 256 different symbols. Certain terminals only support up to 128 different symbols. The national keyboards supplied with these terminals sacrifice some of the symbols (such as `{` and `\`, although these are very useful in the context of the UNIX Operating System) and replace them with local language characters. The terminal itself usually has a Setup key that allows the user to specify the language of choice to make the keyboard function properly.

The substitution characters can still be generated, but not displayed (see “Displaying Data” later in this chapter). To accommodate programmers who use such terminals, a new feature called *trigraphs* has been introduced into the ANSI version of the C language.

Trigraphs are three-letter sequences used in an ANSI C source file that are interpreted as a single symbol (essential to the C language). This allows a programmer who uses an Italian 7-bit terminal, for example, to still get the job done. The one-to-one relationship between trigraphs and the symbols they represent is shown here:

Trigraph	Symbol Represented
??=	#
??/	\
??'	^
??([
??)]
??!	
??<	{
??>	}
??-	~

Note that this feature is not available with the traditional Kernighan and Ritchie (“cc”) C compiler.

Using the VP/ix Environment

The Virtual Personal computer Interactive eXecutive environment (VP/ix) is a UNIX System application that emulates an IBM PC/XT™-compatible computer. It allows users of the INTERACTIVE UNIX Operating System to run DOS and DOS applications as if they were UNIX System utilities. A copy of DOS is furnished with the product and is used by default whenever `vpix` (the name of the actual command) is invoked.

When the VP/ix Environment is used, all previously installed keyboard mapping is automatically disabled until the user leaves the VP/ix Environment. If a non-U.S. keyboard is used, DOS must be informed. With the VP/ix Environment, the system administrator can choose to give each VP/ix user an individual C: drive (this is a virtual disk drive, in reality a UNIX System file, that contains DOS and is used to boot it) or to use a system-wide C: drive. When a non-U.S. keyboard is used, using individual C: drives is preferable. This drive contains the essential DOS system files, `CONFIG.SYS` and `AUTOEXEC.BAT`, that need to be edited to insert information about the keyboard and language used, as well as which country’s conventions should be applied. Refer to the documentation that accompanied your DOS system for details.

Entering Data and Using INTERACTIVE X11

When INTERACTIVE X11 is used with the system, a special program called a `display server` is invoked. This program switches the system from a character-based environment to an all-graphical environment. From that point on, all mapping information specified through the `ttymap` interface is no longer used. The server program is responsible for performing the correct actions each time a key is pressed on the keyboard. By default, it treats any keyboard as a U.S. keyboard. A utility called `xttymap` is provided to change the default actions of the server. It can read and interpret the same input file that is used with `ttymap`.

Due to limitations in the MIT code of X11 Release 4, Compose key sequences and deadkeys cannot be supported when X-based applications are run. The one exception to this, however, is when text-based applications are used in an `xpcterm` window. These applications have access to the tty subsystem, so `ttymap` can then be used to define deadkeys or compose sequences.

Storing Data in the Computer

The previous section explained how keyboards are used to generate letters and other characters on a computer running the INTERACTIVE UNIX Operating System. Typically, these characters are processed by the application that is currently running (it could be the shell, which is the command interpreter, or an editor, or any other application). In most cases, the characters are echoed on the screen.

Applications such as editors, `vi` or `e` (the TEN/PLUS editor), for example, store these characters in a file. As mentioned earlier, a computer speaks no particular language and has no notion of what a letter is. It stores numbers in the file rather than letters. Unless every computer system uses the same number to store a certain letter, files created on one computer cannot be read on another.

Most computer manufacturers use the same convention to represent characters internally; however, some differences in standards do exist.

ASCII

ASCII (American Standard Code for Information Interchange) is a convention, or *codeset*, describing one-to-one relationships between symbols and numbers. It represents letters as numbers that can be stored in 7 bits of the computer's memory, which means a choice of 128 different symbols (0 to 127). The numbers 0 to 32 are reserved for characters that cannot be displayed on the screen but have a special meaning to the system (so-called nonprintable characters). As an example, 7 represents the sound a computer makes when you press Control-g. These characters are often referred to as control characters because the Control key is needed to generate them.

Only 7 bits of internal storage are needed to store 128 different numbers (0 - 127), so the ASCII codeset is called a 7-bit codeset (7-bit US ASCII).

The 96 printable ASCII characters are encoded as follows:

32	SPC	33	!	34	"	35	#	36	\$	37	%	38	&	39	'
40	(41)	42	*	43	+	44	,	45	-	46	.	47	/
48	0	49	1	50	2	51	3	52	4	53	5	54	6	55	7
56	8	57	9	58	:	59	;	60	<	61	=	62	>	63	?
64	@	65	A	66	B	67	C	68	D	69	E	70	F	71	G
72	H	73	I	74	J	75	K	76	L	77	M	78	N	79	O
80	P	81	Q	82	R	83	S	84	T	85	U	86	V	87	W
88	X	89	Y	90	Z	91	[92	\	93]	94	^	95	_
96	`	97	a	98	b	99	c	100	d	101	e	102	f	103	g
104	h	105	i	106	j	107	k	108	l	109	m	110	n	111	o
112	p	113	q	114	r	115	s	116	t	117	u	118	v	119	w
120	x	121	y	122	z	123	{	124		125	}	126	~	127	DEL

There are a few interesting points about the ASCII codeset. Uppercase characters are represented using lower numbers than lowercase characters, and the difference between the value of an uppercase character and its corresponding lowercase character is constant (32). This has often been used (and misused) by programmers. The last character, 127 (Delete), is not always printable. This does not cause any problems, as this character is usually used by the INTERACTIVE UNIX Operating System to interrupt programs. The ASCII codeset contains all letters of the English alphabet and none of the additional letters used in French, German, and other languages.

8-bit Characters and Codesets

Inside the computer, 7-bit numbers are actually stored as 8-bit entities. In most computers, a byte (8 bits or a series of 8 possible zeroes and ones) is the smallest possible unit used to store information, which makes it possible to actually use 256 different characters and symbols. Today this is true if you use the console.

Historically, the eighth bit of the byte that is used to store characters was used by the UNIX Operating System and its utilities for a variety of purposes. It could be used in a sorting algorithm to see if a character was already processed or, when a program allocated bytes of memory, to indicate that the byte was already used. In communication software across telephone lines (which are not

100 percent reliable), the eighth bit was used to do additional checking by forcing the software to use either even or odd values for the number represented by the byte to send across the wire. This bit was then called a parity bit.

Most utilities provided with the UNIX Operating System were careless enough to ignore the value of this last bit, preventing the use of characters with the 8-bit set (such as the ones displayed when running the program listed above), usually referred to as 8-bit characters. Utilities such as `vi` were basically useless for editing non-English texts.

Today, most utilities are what is called “8-bit clean.” The INTERACTIVE UNIX Operating System contains these 8-bit utilities. A notable exception is the `csh` shell.

As 8-bit characters are now supported, an 8-bit codeset can be used, and the convention is to map 256 unique symbols to 256 unique numbers. As might be expected, more than one such codeset exists in the industry. Fortunately, all have one important feature in common: the first 128 characters of these codesets are exactly the same as the characters in the ASCII codeset. In other words, they are all supersets of the ASCII codeset.

IBM Codepages

The codeset used in IBM-compatible personal computers is probably the single most popular codeset used today, primarily by people who are not even aware that it is designed to support non-English languages. Until recently, this codeset was referred to as IBM-extended ASCII (which is a very good description of what an 8-bit codeset is: it extends the 128 character ASCII codeset by another 128 characters).

The characters used in this codeset and the way they are encoded are exactly those characters displayed by the sample program, `show.c`, used in the previous section. If you run this program again and look at the output, you will note the following:

- There is a symbol for almost every code in the second half of this codeset.
- The symbols consist of accented letters, both uppercase and lowercase, special symbols, and graphics characters to draw lines and boxes.
- For some lowercase accented characters, there are no uppercase equivalents (for example, `ë`).

Many personal computer programmers and applications use the graphics characters to draw straight lines, boxes around text, and so on. This codeset clearly supports most characters used in the major Western European languages, such as French and German. In recent years, alternate codesets have been developed for personal computers, and software has been developed to change the codeset used by them when running DOS. (Software to support this was developed for the INTERACTIVE UNIX Operating System as well.) In the DOS world, the name *codepage* was used, and the popular IBM-extended ASCII codeset is now called IBM codepage 437.

The introduction of additional codesets supports more languages spoken in a particular territory. Here are some of the existing IBM codepages and the targeted area or language.

Codepage	Territory or Language
437	U.S. English and Western Europe
850	International codepage (supports more letters and fewer graphics characters than codepage 437)
863	Canada
865	Norway/Denmark
866	Supports Russian alphabet

This list is incomplete; there are codepages for Greek and for the Slavic languages as well. Try running the program from the previous section again, but showing codepage 850 instead. Type:

```
$ loadfont 850
```

The screen will flash and the shell prompt will reappear. Now the console is using a different codeset. Notice the differences between the output of the command and the previous output. To switch back, type:

```
$ loadfont 437
```

ISO Codesets

The organization that sets international standards, called *ISO*, has also defined 8-bit codesets to be used on computer systems in different territories. This standard is more widely adopted on larger computer systems running the UNIX Operating System. This family of codesets is referred to as the ISO 8859 standard. The codeset used in Western Europe is the 8859-1 codeset, which is the standard adopted by the X/Open Company for information interchange. Type:

```
$ loadfont 8859
```

and run the show program again. The following can be observed:

- There is no symbol for the first 32 values of the second 128 numbers.
- There are no graphics characters to draw boxes.
- The difference between the values of an uppercase character and a lowercase character is always constant (32).
- The values chosen for the accented characters are different from IBM codepage 437 (for example, ê is represented by 234 in ISO 8859-1 and by 134 in IBM codepage 437).

To switch back, type:

```
$ loadfont 437
```

There are nine different 8859 codesets, each for a different territory. The most important ones are shown here.

ISO Codeset	Territory or Languages Intended
ISO 8859-1	Western Europe
ISO 8859-2	Eastern Europe (Czech, Polish, and so on)
ISO 8859-5	English and Russian alphabet
ISO 8859-7	English and Greek alphabet

7-bit Codesets

Earlier in this document, we described terminals that support only 128 different characters and use a Setup key to select a language or country. The 7-bit characters generated by most of these terminals follow an ISO standard convention, ISO 646, which is the ISO code name for the ASCII standard. For use with languages other than English, the local language letters are substituted for symbols such as {.

Choosing and Configuring a Codeset

It is the system administrator's responsibility to deal with codesets. The INTERACTIVE UNIX System utility that configures the system to correctly store characters that are generated by the keyboard is the same utility that is used to configure the keyboard, `ttymap`. The system administrator has to verify that data storage happens consistently, regardless of the type of terminal used. Otherwise what was entered as an `ë` on the console yesterday may appear as a `{` on a regular terminal today.

The system administrator must choose between one of the IBM codepages and one of the ISO 8859 conventions. The first issue that determines which one to use is obvious—which language(s) will be used on the system. The other criteria that should be considered in this decision are as follows:

- If many files developed on a DOS system need to be processed or many applications will be used in the VP/ix Environment, an IBM codepage should be used.
- If the system needs to communicate with a heterogeneous network of computers, an ISO 8859 codeset is the better choice.

All of the files supporting international keyboards that are supplied with the INTERACTIVE UNIX Operating System (which are located in `/usr/lib/keyboard`) configure the console to use the IBM codepage 437 (850 for Norway). Additional mapping files are provided as-is with the International Supplement, located in subdirectories of `/usr/lib/keyboard`. They are named after the codeset, 437 or 8859-1, for example, and their names follow the X/Open™ convention for `locale` names, for example:

```
/usr/lib/keyboard/8859-1/fr_FR
```

which represents the mapfile for the French language used in France, using the ISO 8859-1 codeset.

Converting From One Codeset to Another

The International Supplement contains a utility, `iconv`, which can be used to convert the encoding of characters in a file from one codeset to another. The following example shows the command needed to convert the encoding in *filename* from the IBM codepage 437 to ISO 8859-1:

```
$ iconv -f 437 -t 8859 filename > file.new
```

Refer to `iconv(1P)` for more details.

Displaying Data

When characters are displayed on the screen of your terminal or console, these characters physically consist of a set of dots that make up the picture of the character. Typically, a rectangle of 8 by 16 dots is reserved for every character. The one-to-one relationship between a character (actually the numeric representation of a character) and its picture is called a font. Depending on how the INTERACTIVE UNIX System is used, fonts may or may not be modified.

After typing a character and possibly storing that character in file, a code (usually the same as the input code) is sent to the terminal to indicate that it should display something. If necessary, the code sent by the system or the application can be modified before it is sent to the screen. This practice is called *output mapping*. Again, `ttymap` is the utility responsible for this function. Proper output mapping and possible modification of the font guarantee the display of the proper character (or, when the actual character cannot be displayed, at least something that makes sense). Here are a number of suggestions for making the INTERACTIVE UNIX System work correctly.

7-bit Terminals

When 7-bit character terminals are used, a 128-character font that is hardcoded inside the terminal hardware is used. This font cannot be modified, but more sophisticated terminals allow access to several different fonts, one for each

language supported. These terminals support the ISO 646 ASCII variants described in the previous section. To ensure consistency throughout the system (assuming a French 7-bit terminal is used):

- On input, map the 7-bit code generated for the French characters into their actual 8-bit value.
- On output, map the 8-bit code back to the 7-bit code to display the correct French character.
- Use trigraphs for ANSI C programming.
- To generate curly braces and other such characters, use the decimal representation. On output, map to a space character.

This ensures the proper display of the file used, especially when the same file is later edited on devices such as the console.

If the inability to display curly braces and other typical UNIX System characters, such as `\`, is too annoying, use this alternative approach:

- Use the Setup key of the terminal to switch to U.S. English. You now have access to a U.S. ASCII font but still have a French keyboard layout.
- When a French character key is pressed, it is mapped and stored using its correct 8-bit value.
- On output, it is mapped to the corresponding character without the accent, or the closest-looking English letter (for example, a “c” instead of a ç).
- Use decimal representation for the UNIX System characters, which are automatically stored as 7-bit characters and displayed correctly.

Your system administrator should develop the correct `ttymap` description file for your machine.

The Console

On the console, a font of 256 different symbols can be used. That font information is stored in Random Access Memory (RAM) on the video card inside the computer, to which the monitor is attached. The information can be changed (on old or inexpensive systems, the information is stored in Read Only Memory (ROM) and can only be changed by replacing the ROM with a different ROM).

The `loadfont` utility changes the font information in the video card. This utility has predefined, built-in fonts. However, anyone can use it to develop a personalized font. Refer to *loadfont(1)* for more information.

Displaying Data and Using INTERACTIVE X11

INTERACTIVE X11 and X11-based applications always use fonts when text is displayed. Most applications have a command line option, `-fn`, to indicate which font to use. Fonts for both the 8859-1 (most of the supplied fonts) and IBM 437 codesets are supplied with INTERACTIVE X11. The font files supplied with the International Supplement can also be used with INTERACTIVE X11 after converting them with the `bdf2osnf` utility.

The International Environment

Running applications in an internationalized environment is based on the concept of a local environment or `locale`, which is defined as the subset of the user's environment that depends on language and cultural conventions.

A `locale` consists of a number of categories, with each category controlling a specific aspect of the international environment. Each category is usually referred to by the variable used to set or modify it. The International Supplement recognizes the following categories:

Date and Time Format

This category, `LC_TIME`, affects how date and time are displayed.

Character Classification

This category, `LC_CTYPE`, defines codeset characteristics and character classification.

Collation

This category, `LC_COLLATE`, affects the collation ("sorting") order.

Numeric and Monetary Formatting

These categories, `LC_NUMERIC` and `LC_MONETARY`, affect the format of nonmonetary and monetary numeric information, such as the decimal delimiter.

Yes/No Responses

This category, `LC_MESSAGES`, affects the strings used to indicate yes/no answers to utility and application queries. (Note that while the

internationalized yes/no response is required by XPG for certain commands, the LC_MESSAGES category is not part of the locale as defined by XPG.)

Message Catalogues

Message catalogues are not yet covered by the locale categories, but use similar mechanisms.

Internationalized Behavior

This section explains how the international environment affects the behavior of system utilities and applications.

Date and Time Format

The default conventions for the date and time format, as well as the names of the days of the week and months, follow U.S. conventions and are rarely applicable in other countries. By defining and using the date and time environment, the dates and times displayed by the system, utilities, and applications follow the local conventions and use the names of the days and months in the correct language.

The following aspects of formatting are supported by the INTERACTIVE UNIX Operating System:

- Format of time display
- Format of date display
- Format of combined date and time display
- Format of 12-hour time display
- Names of days of the week
- Abbreviated names of days of the week
- Names of the months
- Abbreviated names of the months
- Format of the ante meridiem and post meridiem strings used in 12-hour clock time displays

For example, in a French environment, the output of `date` could be:

```
Mardi 30 juillet 1993 11:07:35 PDT
```

and the output of `ls -l`:

```
total 636
-rw-r--r-- 1 paul  other 27399  janv.  24 18:36:02  ch01
-rw-r--r-- 1 paul  other 13842  juil.   9 18:36:03  ch02
-rw-r--r-- 1 paul  other  9057  mai    12 18:36:03  ch03
-rw-r--r-- 1 paul  other   263  mai    12 15:44:45  document
-rw-r--r-- 1 paul  other   398  sept.  24 12:37:34  Makefile
-rwxr-xr-x 1 paul  other 24202  avril  10      1991  show
```

Utilities will check for the value of the environment variable to determine which format should be used. See `date(1)` for more details.

Character Classification

Regardless of how it is encoded, a character has certain features. For example, it is either printable or nonprintable. If a different codeset is used, different numbers represent the characters. To keep track of this, the system uses a classification table, which contains information about all 256 possible characters in the codeset. Information that can be specified are:

- Lowercase letters
- Uppercase letters
- Digits
- White-space characters
- Punctuation characters
- Control characters
- Uppercase to lowercase conversion
- Lowercase to uppercase conversion
- Printable characters or nonprintable characters

Programs that are written to use functions like `isupper` and `isdigit` (refer to `ctype(3C)`) access this table and behave accordingly. The default table used by the system is the ASCII table that considers every 8-bit character (that is, with a value greater than 127) to be nonprintable. This explains why programs

such as `vi` do not display 8-bit characters correctly, but their octal representations instead, unless the proper environment is set up. That environment is set by using the variable `LC_TYPE`.

The INTERACTIVE UNIX Operating System fully supports internationalized regular expressions. Where appropriate, UNIX System utilities have been enhanced to support these capabilities. These utilities are supplied with the International Supplement subset. For a detailed description of internationalized regular expressions, refer to *regexp(5P)*.

Collation

Collation, according to a dictionary, is the “act of putting things in their proper order.” Thus, collation rules define how the data are put in the proper order, or *sorted*. Traditionally, the collating order in the UNIX System has been ASCII order, that is, the order in which the characters appear in the ASCII codeset. This is the natural collating order for the English language.

For most languages in the world, however, this is not enough. Most European languages contain more letters than the 26 in the English language, with the additional letters typically collating between the letters in the ASCII set. For instance, an accented `á` sorts between `a` and `b`. Most European users expect sorted lists (for instance, the output from the `ls` command) to appear in the collation order of their language.

Languages with non-Latin-based alphabets, such as Russian or Greek, use a completely different set of characters. For these languages, collation takes on additional complexities.

The INTERACTIVE UNIX Operating System allows users to define their own collation order. The standard utilities that depend on collation, such as `sort` and `ls`, have been modified to understand this user-specified collation order and are supplied with the International Supplement subset.

An Example

Consider the following four lines (the four seasons in French):

```
printemps
été
automne
hiver
```

The regular UNIX System `sort` utility sorts them as follows:

```
automne
hiver
printemps
été
```

It uses the numeric representation of characters, and because `é` is represented by an 8-bit character, it is listed last. The UNIX System `sort` formerly stripped the eighth bit, sorting the above sequence as:

```
été
automne
hiver
printemps
```

which is, of course, wrong as well. (This illustrates that simply making utilities 8-bit clean is not always sufficient.)

The internationalized `sort` gives the following (correct) result:

```
automne
été
hiver
printemps
```

Numeric and Monetary Formatting

The default conventions for decimal delimiter and other numeric formatting rules are seldom correct in an international environment. For example, the default decimal delimiter in the U.S. is a period, but in most European countries the comma is used instead, which, in turn, is used in the U.S. as the thousands separator character. So \$1,000, which is one thousand dollars in the U.S., could be interpreted as a single dollar in Europe. Misinterpreting things the other way around could be quite an expensive mistake! By defining numeric and monetary formatting with the correct values, programs display fractions using the appropriate decimal delimiter.

Applications such as accounting programs often have to be modified to display the correct monetary symbol. The manner in which numbers representing amounts of money are formatted is also subject to local conventions.

Yes/No Responses

Some utilities, such as `rm`, require the user to acknowledge whether a specific action should be taken. The usual response is either “yes” or “no.” Before internationalization, such utilities required the user to respond using the English `y` or `n`. Such a response is not natural to French-speaking people in the world, where, of course, `oui` would be more natural instead of `yes`. The INTERACTIVE UNIX System includes; the capability to define the correct yes and no responses for a particular `locale`.

Part 2 — System Administration for New Users

This part of the book discusses some of the basic procedures required to keep your INTERACTIVE UNIX Operating System running smoothly. It also provides tutorial information for performing some fundamental system administration tasks.

Part 2 — “System Administration for New Users” is intended for individuals responsible for running an INTERACTIVE UNIX System who have little or no experience with any UNIX-based operating system. It does *not* provide complete instructions for all of the system management procedures required to maintain the operating system.

This part of the book supplements the technical information found in the *INTERACTIVE UNIX System Installation Guide*, which provides information on installing the INTERACTIVE UNIX System and optional software. The *INTERACTIVE UNIX System Maintenance Guide* also provides more detailed technical information on configuring and maintaining an INTERACTIVE UNIX System than is found in this book. However, the material presented here does cover important information not presented in the *INTERACTIVE UNIX System Maintenance Guide*, such as creating user accounts, and backing up and restoring files.

What Is a System Administrator?

The *system administrator* is responsible for installing, administering, and maintaining your INTERACTIVE UNIX Operating System. If the INTERACTIVE UNIX System is installed on a personal computer, you will probably act as the system administrator. If your personal computer is part of a larger network of computers, a system administrator from the computer operations department of your company may be assigned to help you keep your system running smoothly. This document assumes that *you* will be responsible for basic system administration tasks.

A system administrator can have many responsibilities, but there are several fundamental tasks that must be performed after your system is initially installed. You will be responsible for:

- Managing user accounts
- Tailoring the system
- Installing new software packages
- Installing new hardware devices and their associated drivers
- Maintaining file systems
- Performing routine maintenance procedures, such as backing up files

You should use the *console terminal* to perform all system administration tasks. The console terminal is comprised of the screen and keyboard directly attached to the Central Processing Unit (CPU). It is important for the system administrator to use the console terminal since most error messages generated by the system are displayed on the console terminal screen.


The following chapters provide you with a conceptual overview of each of the topics listed above. Where appropriate, tutorial instructions are included. You may also want to refer to the *INTERACTIVE UNIX System Installation Guide* and the *INTERACTIVE UNIX System Maintenance Guide* for more information when you are using your computer.

Before You Begin

Before you attempt any of the system administration procedures, you should understand the material in Part 1 — “INTERACTIVE UNIX System Primer.” You will also need to be familiar with a text editor if your particular hardware and software configuration requires you to tailor some of the standard configuration files. You may also need to refer to the *INTERACTIVE UNIX System Installation Guide* and the *INTERACTIVE UNIX System Maintenance Guide* or to any additional documentation supplied by your vendor to completely tailor your system.

Remember that manual entries are referenced using the form *entryname(n)*, where *entryname* is the name of the entry and “n” is the number of the section. Refer to the manual entries as often as necessary.

Shutting Down and Bringing Up the System

16 

The procedures for shutting down and bringing up the system should already be familiar to you if you were responsible for installing the system, as outlined in the *INTERACTIVE UNIX System Installation Guide*.

Shutting Down the System

When you are ready to turn off your computer, you must arrange to have the computer complete all of the tasks that are currently **running**. This is accomplished with a system maintenance procedure called `shutdown`. The `shutdown` program gracefully terminates all tasks that are currently executing before it halts the system. When `shutdown` has finished, you may safely turn off the computer. If you do not run the `shutdown` program, you may lose data that is stored on your system and cause damage to your file system, even if no one is currently logged in.

The `shutdown` program can be run in two ways:

- Using the `powerdown` administrative login
- Using the `shutdown` command

Using the powerdown Administrative Login

When you are ready to turn your computer off, you may use the powerdown administrative login to bring the system down.

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

Note – You must remember the powerdown password you set during installation, or use the `passwd` command to set a password for the powerdown login.

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:
INTERACTIVE UNIX System V/386 Release 3.2, Version 4.1
amadeus
Copyright (C) 1994 Sun Microsystems, Inc.
Copyright (C) 1988 AT&T
Copyright (C) 1988 Microsoft Corp.
All Rights Reserved
Login last used: Thu Apr 7 20:31:37 1994
/          :      Disk space:  82.84 MB of 100.00 MB available(82.84%)
/home      :      Disk space: 222.46 MB of 226.00 MB available(98.44%)

Total Disk Space:          305.30 MB of 326.00 MB available(93.65%)

Once started, a powerdown CANNOT BE STOPPED.
Do you want to start an express powerdown [y, n, ?, q]
```

4. To bring the system down, type `y`. The system responds:

```
Shutdown started.      Thu Apr  7 20:33:37 PDT 1994

Broadcast message from root (console) on amadeus Thu Apr  7 20:33:38...
THE 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.
cron aborted: SIGTERM
! SIGTERM Thu Apr  7 20:33:43 1994
! ***** CRON ABORTED ***** Thu Apr  7 20:33:43 1994

The system is down.
Press any key to reboot.
```

5. When the Press any key to reboot message appears, the computer can be safely turned off.

Using the shutdown Command

To execute the shutdown command manually, you must log in to the system as the *root user* on the console terminal and you must be in the *root* directory.

Note – The *root* user is the most privileged user on the system; this login ID is commonly known as the *superuser*. There are no restrictions imposed upon *root*, which means that the person logged in to the system with the *root* ID can access, modify, and delete every file and process on the system. The *root* login is discussed in more detail in Chapter 17, “Special User Accounts on the INTERACTIVE UNIX System.” When you are logged in as *root*, your login prompt is a pound sign (#).

To execute the shutdown command manually:

1. Log in as root. Your screen will look similar to this:

```
login: root
Password:
INTERACTIVE UNIX System V/386 Release 3.2, Version 4.1
amadeus
Copyright (C) 1994 Sun Microsystems, Inc.
Copyright (C) 1988 AT&T
Copyright (C) 1988 Microsoft Corp.
All Rights Reserved
Login last used: Thu Apr 7 20:31:37 1994
#
```

The # prompt indicates that the system is ready for you to type in a command.

2. Move to the root directory by typing:

```
# cd /
```

Refer to Chapter 6, “INTERACTIVE UNIX System Directory Structure,” for more information about directories.

3. Run the shutdown program by typing:

```
# shutdown
```

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

```
Shutdown started. Thu Apr 7 20:03:35 PDT 1994

Broadcast message from root (console) on amadeus Thu Apr 7 20:04:36...
THE SYSTEM IS BEING SHUT DOWN NOW ! ! !
Log off now or risk your files being damaged.
```

The system gives users 1 minute to exit editors and save files. After this 1 minute warning period is over, the shutdown program prompts you for confirmation that you want to continue. Your screen will look similar to this:

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

4. Type *y* if you want to continue shutting down the system, *n* if you want to stop.
 - If you type *y*, the system is then shut down. You may press any key to reboot the system or turn off the power to the computer at this point.
 - If you type *n*, the system issues the following message to all users currently logged on to the system:

```
Broadcast message from root (console) on amadeus Thu Apr 7 20:07:44...  
False Alarm: The system will not be brought down.
```

The system then aborts the shutdown and displays the following message on the console:

```
Shut down aborted.
```

5. If you want to give users a longer warning period before the system goes down, run the shutdown program using the *-g* option:

```
# shutdown -gtime -y
```

In this example, *time* represents the number of seconds the system will wait before it goes down. It is a good idea to allow at least 2 minutes (120 seconds) to elapse before the system is brought down. Using the *-y* option causes the shutdown procedure to continue directly without prompting you for permission to proceed.

After the warning period, the system automatically runs shutdown. Your screen will look similar to the one generated by the powerdown procedure:

```
Shutdown started.    Thu Apr  7 20:33:37 PDT 1994

Broadcast message from root (console) on amadeus Thu Apr  7 20:33:38...
THE SYSTEM IS BEING SHUT DOWN ! ! !
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.
cron aborted: SIGTERM
! SIGTERM Thu Apr  7 20:33:43 1994
! ***** CRON ABORTED ***** Thu Apr  7 20:33:43 1994

The system is down.
Press any key to reboot.
```

Be sure that you do not attempt to power down or reboot the system until the above completion message is displayed.

See *shutdown(1)* for more information.

Bringing Up the System

To bring up or *reboot* the system, use this procedure:

1. First check the diskette drives and remove any diskettes there. If the system finds a diskette when it is first turned on, it will attempt to boot from it. If it does not find a diskette, it will boot from the fixed disk when the power is turned on.
2. If your computer is still turned on, either turn it off and turn on the power again, or press any key to reboot.
3. If your system power is turned off, turn on the power. The INTERACTIVE UNIX Operating System automatically boots.

When it reboots, the system does the following:

- Runs internal diagnostic programs to check your hardware
- Displays memory and copyright information and the message:

Booting the INTERACTIVE UNIX Operating System

- Determines if the system was shut down properly
- Checks the file systems for inconsistencies and attempts to make any necessary repairs
- Starts various system processes, such as the line printer scheduler

If your system has crashed for some reason, the system may ask if you want to save a *system dump*. A system dump is used by experienced system administrators to obtain detailed information about the state of the system when it crashed. Under most circumstances, you will want to answer *n* to this prompt.

Note – You can configure the system to not ask about saving a system dump by uncommenting the `SAVEDUMP=no` line in `/etc/default/boot`.

Special User Accounts on the INTERACTIVE UNIX System

17 

There are three types of login accounts on an INTERACTIVE UNIX Operating System: an *ordinary* login account, a *system* login account, and an *administrative* login account. A login account gives access to the system.

Ordinary logins include a login ID, a password, and an area of the file system “assigned” to the user, called a *home directory*. Every user on the system (including the system administrator) uses an ordinary login account to perform most tasks. These accounts are used to store data and run application programs, such as an accounting or word processing program. Ordinary logins may have restricted access to other user login accounts and to system administration functions. The procedure for creating an ordinary user account is described in Chapter 19, “Managing User Accounts.”

System logins are used to perform system administration tasks that require privileged access to the restricted files and directories on the system. You typically use a system login when you need to perform system maintenance tasks for the operating system or hardware facilities on your system. These logins should be used with great care.

Administrative logins are used to give ordinary users restricted access to system management functions that must be performed frequently. For example, there are administrative logins available for adding new users or shutting down the system.

Note – The system is delivered with all system and administrative accounts, except the `root` account, locked. The system administrator should *immediately* set a password for the `root` account and should then use the account to change the password for all the other system and administrative accounts.

System Login Accounts

When you initially installed your system, several system logins were automatically created for you. The two most important system logins are `root` and `bin`. They are used to perform most privileged system administration functions. Other system logins are available to monitor and manage continuously running processes, to run the program that updates the fixed disk, and to manage networking operations.

Note – All system logins should be used *only when absolutely necessary*.

System logins have few restrictions imposed upon them by the operating system, which means that they can cause serious damage to your system when used incorrectly. A password for each of the system logins, except `root`, must be assigned before that login can be used. (See the *INTERACTIVE UNIX System Installation Guide* and the *INTERACTIVE UNIX System Maintenance Guide* for more information about assigning system passwords.) Passwords for system logins should only be given to the individual(s) designated as the system administrator(s) for your system. Each system login is described below.

The root Login

The most important INTERACTIVE UNIX System login belongs to the *root user*. This login account, also known as the *superuser*, is the most privileged and powerful account on the system. There are *no restrictions* imposed upon `root`, which means that individuals who have access to the `root` login also have access to every other account on the system. The `root` user can access, modify, and delete every file and process on the system.

It is very important to:

- Assign a password to the `root` account.
- Change the `root` password frequently.

- Give it only to those performing system administration tasks.
- Use it *only* for system administration tasks.
- Be very careful when using it.

You must be at the console terminal to log in to the system as `root`. This is an additional safeguard that helps limit the use of the `root` login. You will use the `root` login to configure the system kernel, add new peripheral devices (such as printers), and mount or unmount file systems.

Once you have logged in to the system using the `root` login, the system displays the pound sign (`#`) as the prompt.

The bin Login

The `bin` system login owns most of the command files and directories on the system, including all of the special files used to access fixed disks and diskettes. The `bin` login account has the same privileges as an ordinary user, but also owns the directories `/bin`, `/usr/bin`, `/lib`, and `/etc`. You must log in as `bin` (or be logged in as `root`) to install new programs and libraries in these directories. You can use the `bin` login to install new software or to modify the files and directories owned by `bin`. This login is also used to limit user access to files and directories that can affect the entire system. You will not normally need to use the `bin` login.

Other System Logins

The system automatically provides several other system logins: `daemon`, `sync`, `uucp`, and `nuucp`. You should assign a password to these accounts, but you should never need to log in using `daemon`, `sync`, or `nuucp`. These system logins and their functions are described below.

`daemon`

A *daemon* is a program that executes continuously on the system. For example, the program responsible for queuing print jobs is a daemon. This login is provided because certain programs on the system must be owned by `daemon`.

`sync`

This login executes the `sync` program, which updates the fixed disk with any work currently in progress.

uucp

This login owns the files in the directory `/usr/lib/uucp`. It may occasionally be necessary to log in as `uucp` to change these files, which are used in the UNIX-to-UNIX-copy file transfer programs.

nuucp

This login is used by remote machines to log in to your computer system and to initiate UNIX-to-UNIX file transfers.

Administrative Login Accounts

There are a number of tasks that require privileged access to the system. Because the `root` login is so powerful, it is not a good idea to use it to perform all of the administrative tasks that are required to keep the system running properly. Instead, several administrative logins are provided with your system to facilitate system maintenance. As with all sensitive login accounts, you must set a password for each login and limit the password's distribution. The administrative login accounts are described below.

sysadm

This login account gives you direct access to the `sysadm` facility. `sysadm` is a menu-driven facility that can be used to perform most system administration procedures. It is described in Chapter 18, "The System Administration Program."

powerdown

The `powerdown` login can be used to shut down the system. All the processes running on an INTERACTIVE UNIX System must be gracefully terminated before the computer is turned off, in order to prevent serious damage to the file system or loss of data. The `powerdown` login is provided to facilitate this process. Refer to Chapter 16, "Shutting Down and Bringing Up the System," for more information.

checkfsys**makefsys****mountfsys****umountfsys**

These four logins may be used to directly access the file system maintenance options that are available as part of the `sysadm` login and *utility*. (A utility is a program, usually from a set of programs, that

represents a specific application available with your computer.) You can use the `sysadm` login to access these options as well. Refer to Chapter 18, “The System Administration Program,” and Chapter 24, “File System Maintenance,” for more information.

Accessing Other User Accounts (`su`)

If you know another user’s login name and password, you can use the `su` command to temporarily switch to that user’s account and gain access to protected files, directories, and programs. This can be very useful, for example, if you need to temporarily access either a system or an administrative login without logging out. To access another user account without logging out, follow these instructions (this example shows how to access the `root` account using `su`):

1. Use the `su` command to acquire `root`’s privileges without logging out. (Note that although you have to be at the console terminal to *log in* as `root`, you do not have to be at the console terminal to `su` to `root`.) Type:


```
$ su - root
```

and press Enter.

2. If a password is assigned to that account, the system asks for a `Password:`. Type the correct password and press Enter.
3. The system responds with the `#` prompt. You now have all the privileges of the `root` user, and you are placed in `root`’s home directory, `/`.
4. Type `exit` or use Control-d to exit the `root` account and return to your normal user account.

This same procedure can be used to access any user account for which you know the correct login name and password.

The System Administration Program

18 

The system administration program is a standard feature of the INTERACTIVE UNIX Operating System. It is a menu-driven interface that makes it easy to perform commonly required system administration tasks and is designed to be as self-explanatory as possible. The `sysadm` program makes use of the Character User Interface (CUI). If you have never used `sysadm` before, you should read, either in this guide or on-line, the short summary of how to get help and how to use the menus and forms to perform tasks.

This chapter provides an overview of the system administration menus and the purpose of each. The use of specific system administration procedures is discussed in the appropriate task-related sections of this document and in the *INTERACTIVE UNIX System Maintenance Guide*.

Note – This chapter does not attempt to describe every possible maintenance procedure available through the system administration menu. As you become familiar with this interface, you may wish to explore the usage of some of the menu items that are not described here by simply trying them out. The system prompts you step-by-step through the procedures.

`sysadm` is both a login account and a utility program. This means that you can access `sysadm` either by logging in to the system with the `sysadm` login or by typing the `sysadm` command when you are already logged in to the system. If you log in as `sysadm`, the system displays a screen similar to this:

```
login: sysadm
Password:
```

You must know the `sysadm` password to use this login. If you are already logged in, type `sysadm` at the shell prompt:

```
$ sysadm
```

The system will prompt for a password.

The sysdemo Login

The `sysdemo` login account is a tutorial login that provides the full functionality of the `sysadm` program without actually performing the functions. This provides a safe way to learn to use the system. The `sysadm` forms are brought up on the screen and the hypertext help is available, just as they would be under the `sysadm` account. If you would like to explore the system using the `sysdemo` login, you must know the `sysdemo` password. Your screen will look similar to this:

```
login: sysdemo
Password:
```

If you are already logged in, type `sysdemo` at the prompt:

```
$ sysdemo
```

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 Escape. Table 18-1 summarizes how to use the help system.

Table 18-1 Using the CUI 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 <i>not</i> in the help system
Scroll through the Help Index	Up and down arrow keys
Page through the 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	Escape

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. Press Tab or Back-Tab to move from link to link. When the one you want is highlighted (active), press Enter to follow it.

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 then 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.

Using CUI Menus and Forms

The System: 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.

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.

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, move to `Quit` (or type `q`) and press Enter, or simply press Escape.

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 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.

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.

[]

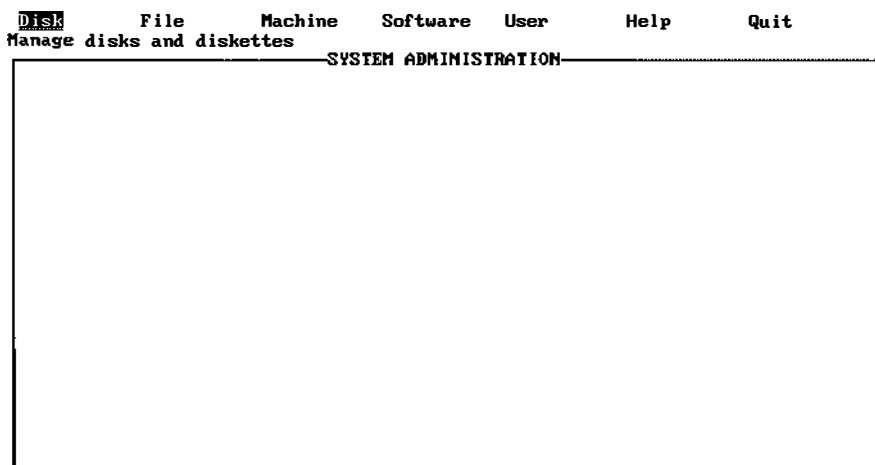
Fields within square brackets [] are check boxes. Press the spacebar to “check” the box (an X appears). You can press the spacebar to toggle the X in or out of a field.

()

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 choices automatically go “off.”

Bar Menu Options

When you type the `sysadm` command, the system administration bar menu is displayed at the top of the screen. If you set a password for the `sysadm` login when you installed the system, you are prompted for a password as usual. The system displays:



Each title on the bar menu represents a category of tasks that are usually performed by the system administrator. When you select an option, the system displays a menu that lists the tasks that are specific to that option. When you select an option from a menu, the system either provides a form with step-by-step procedures for performing the specified task or displays another pull-down menu.

The following menus are available:

Disk Menu

The Disk menu is used for tasks related to your fixed disks and diskettes. The tasks you can perform using the options on this menu include copying diskettes, checking the integrity of diskette-based file systems, deleting data from diskettes, and making new diskettes accessible to the system. You can also use the options on this menu to perform similar tasks on a fixed disk, to inform the system about

defects on your fixed disk, or to determine the current partition configuration of your fixed disk.

Refer to Chapter 24, "File System Maintenance," for information about using the **Disk** menu and to the *INTERACTIVE UNIX System Maintenance Guide* for information about manipulating file systems on your fixed disk.

File Menu

The **File** menu options allow you to back up, restore, configure, and monitor file systems. Use the options on this menu to back up data from your fixed disk onto a diskette or other device or transfer the data to a fixed disk. Refer to Chapter 25, "Backing Up Files," for information about using the **File** menu.

Machine Menu

The **Machine** menu includes options to add a printer to your system and to change the number of virtual terminals available at the console terminal. Other options allow you to initialize your system the first time you install it or to change the current administrative and system passwords, date, and time. In addition, this menu has an option to configure the lines that connect your terminal, modem, or other device to the main CPU.

Software Menu

The **Software** menu options allow you to list, install, configure, and remove software packages and extensions, as well as configure the standard networking utility available for the **INTERACTIVE UNIX System uucp (unix to unix copy)**.

The installation of software packages is discussed in the *INTERACTIVE UNIX System Installation Guide*. See the *INTERACTIVE UNIX System Maintenance Guide* for more information on the installation and configuration of **uucp**.

User Menu

The **User** menu options allow you to establish and configure user login accounts. Among other things, it can be used to list, add, and delete users and groups. This menu is described in more detail in Chapter 19, "Managing User Accounts."

Using the sysadm Menus

The following commands and 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 the `sysadm` password.
- Use the left and right arrow keys `→ ←` to move from title to title of the bar menu. 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`.
- Use the up and down arrow keys `↑ ↓` to move from option to option on a pull-down menu, 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`. To exit from a pull-down menu or return to a previous menu at any point, move to `Quit` (or type `q`) and press `Enter`, or simply press `Escape`.
- Forms are used to display information needed to perform tasks. Forms contain several different kinds of fields. Some fields are used only to display information; you cannot `Tab` or `Back-Tab` 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. `Tab` or `Back-Tab` to move from field to field and button to button. All forms have at least one button in a box at the bottom of the form. To activate a button, `Tab` or `Back-Tab` to it and press `Enter`. Most forms have these buttons:

OK

Confirms that you are satisfied with the information on the form and want your changes, if any, to take effect.

CANCEL

Exits from the form without any changes taking effect.

HELP

Provides help for the form.

To choose a button, `Tab` or `Back-Tab` to it and press `Enter`.

- If you do not understand the options or what to put into a field displayed by the system, press the `F1` key for on-line help at any time. If context-specific help is available, it appears. Pressing `F1` a second time accesses the Help Index. This is a list of topics for which help is available. If no context-

specific help is available, the Help Index appears immediately. Tab or Back-Tab to a topic and press Enter to view a help screen. Use Page-Up or Page-Down for those help topics that contain more than one page.

There are also HELP buttons at the bottom of many forms. Tab or Back-Tab to the HELP button and press Enter to obtain general help on the form you are viewing. Use Page-Up or Page-Down for those help topics that contain more than one page. You can exit from the help system at any time by pressing Escape.

- To exit the `sysadm` program, move to the `Quit` option on the bar menu at the top of your screen and press Enter.
- If your screen is ever obscured by extraneous text, such as console messages, type `Control-r` to refresh it.

sysadm Menu Bypass

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

```
# sysadm disk
```

to access the `sysadm` Disk menu.

Each `sysadm` pull-down menu contains a list of options and/or other menus. You can 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 for formatting a disk, called `Format a Diskette`. You can access that option directly by typing:

```
# sysadm fmtdisk
```

If you use `sysadm` as a login, you can also supply the `sysadm` options described above, such as `fmtdisk`, at the login prompt:

```
login: sysadm fmtdisk
```

Note – See the *INTERACTIVE UNIX System Maintenance Guide* for a complete list of all the `sysadm` menu bypass commands.

The INTERACTIVE UNIX Operating System requires that users log in to the system with a login name and, optionally, a password. This enables the system administrator to keep track of users on the system, control access to the system, and control system resources. All individual users should use a private login. (Other computers can use generic logins such as mail or nuucp to log in to your computer and exchange information.)

The system administrator is responsible for establishing and maintaining a login account for each user who requires access to the system. This includes adding users, aging user passwords (if desired), changing user passwords (if necessary), and removing user logins that are no longer needed. Use the `sysadm` command or `login` to establish new user accounts. `sysadm` provides a safe and secure way to manage new and existing user accounts.

What Is a User Login Account?

A user login account defines the user's environment, which can be very restricted or highly privileged. It is defined by a number of attributes (characteristics) that are established when a login ID is created, and it determines the security levels that are imposed upon the user, including the programs and files that are accessible.

At the simplest level, a user login consists of the following information stored in the `/etc/passwd` file, which defines a user's individual environment:

Login ID

The name used to log in to the system. The user types this name at the `login:` prompt. This one to eight character string must be unique for each user; a first name, last name, initials, or some combination are typically used. It should only consist of lowercase letters and/or numbers. Uppercase letters are permitted, but discouraged, as most UNIX commands expect lowercase letters. Special symbols are not permitted.

Password mark

A mark indicating whether or not the user has a password. An `x` is placed in this field if there is an encrypted password entry in `/etc/shadow` for this user. If the user does not have an encrypted password entry in `/etc/shadow`, this field will be blank. The actual password is a string the user must know and type in when logging in to the system. It should be known only to the user.

User ID

A **user identification number** (frequently abbreviated UID) is a unique number assigned to each login account. The system starts with a default number (for example, 100), then increments the number by one each time a new user is added to the system. This number is associated with all files and processes created by that user name. The user ID is usually unique for each user login.

Group ID

A **group identification number** (frequently abbreviated GID) is a number that associates a user with a group. There may be one or more groups on the system. A default group and group ID number are automatically assigned to a user. An affiliation with a group means that a user can access the files and directories that are owned by members of that group, if the group permissions on those files or directories permit it. The GID is associated with an entry in the file `/etc/group`.

User name

This is the full name of the user (for example, Joe Smith, Jane Grey). It is included because user login IDs are sometimes cryptic.

Home directory

The full path name of the directory into which the user is automatically placed after logging in. This directory usually belongs solely to the

user, who can restrict others' access to the files and directories created there. (Only the user and the superuser can access this directory if the user chooses to restrict all others.) The home directory should be unique for each user login.

Login program name

This is the full path name of the program the system runs each time the user logs in. This is usually the shell (`/bin/sh`), but it may be another program, such as a restricted shell (`/bin/rsh`), the C shell (`/bin/csh`), the Korn shell (`/bin/ksh`), or an application program, such as the VP/ix Environment.

System Files That Define the User's Environment

The /etc/passwd File

The `/etc/passwd` file identifies each user to the system. Each time you add a user to the system, an entry is added to this file. When a user attempts to log in to the system, the system checks the login ID against the list of authorized users that is stored in the file `/etc/passwd`. The user's password is checked against the entry for that user in `/etc/shadow`. If a login ID or password is missing or incorrect, the user cannot gain access to the system. If the user provides a correct login ID and password, the user is logged in to the system with access to all of the resources assigned to that login ID. A password provides an additional level of security on the system and should be set for each user account. By default, the system forces users to establish one the first time they log in, if the system administrator has not already assigned one. (To change the default, refer to *login(1)*.)

Use the `sysadm` command to establish each of these fields for a new user or to change the information in these fields for an existing user account. `/etc/passwd` is automatically modified by `sysadm` each time you add, delete, or change a login account. You should never directly edit the password files themselves. Refer to "Managing User Accounts With `sysadm`" later in this chapter for more information about using the `sysadm` utility.

The `/etc/group` File

Just as `/etc/passwd` defines the individual user's environment, the `/etc/group` file defines the environment for each system group. This file is used to establish another level of security for each login ID. Each user on the system is assigned to one or more groups. Members of a group have access to files and directories that are owned by other members of that group. The default group identity for users of the INTERACTIVE UNIX Operating System is `other`. Additional group identities supplied with your system are `root`, `bin`, `sys`, `adm`, `mail`, and `daemon`. You can create new groups based on your use of the system. The file `/etc/group` contains a one-line entry with the following information for each group:

The group name

A group name is a word, number, or abbreviation that is up to eight characters in length. You typically use a name that refers to the group. For example, you may select `mktg` as the group name for all members of the Marketing Department or `fin` for the people in your Finance Department. A group name can only consist of lowercase letters and/or numbers. Uppercase letters and special symbols are not permitted.

A password

Not used.

A group ID number

This is a numeric identification that is associated with the group name. This number is the same as the GID number found in `/etc/passwd`.

A list of users in the group

You may include a list of user login IDs that are associated with the named group. To include a list of users in this group, separated by commas, you must edit the file directly rather than using the `sysadm` procedure. This entry is optional.

`/etc/group` is automatically modified by `sysadm` each time a new group is added to or deleted from the system.

The /etc/shadow File

Another file, `/etc/shadow`, is used primarily to provide tighter security. It contains the user's actual encrypted password and password aging information. Every time you log in, the system checks your password against your entry in `/etc/shadow`. A password provides an additional level of security on the system and should be set for each user account. Only users with access to the password for a login ID (and the superuser) can access that account. The password is never displayed on the screen. It does not appear when it is first established, when you use it to log in, or when it is changed with `sysadm` or the `passwd` command. This procedure helps to maintain a high degree of system security.

`/etc/shadow` is automatically modified by `sysadm` each time a new user is added to or deleted from the system.

Managing User Accounts With sysadm

Use `sysadm` to manage the facilities that govern user logins. The User menu provides many of the options you need to add, delete, or modify entries in `/etc/passwd` and `/etc/group`. If you are not an experienced system administrator, it is best to use `sysadm` to make changes; do not attempt to modify these files directly.

You are now familiar with the fields that are used to establish login and group IDs. The `sysadm` forms and help screens provide step-by-step instructions that make it clear how to modify one or more of these fields. The `sysadm` User menu includes several options for managing users and groups on the system. You can add a group to the system, add a user to the system, delete a group from the system, delete a user from the system, list the groups on the system, list the users on the system, modify the defaults used by the `adduser` command, modify a user's login, and change a group's name.

After stepping through the two most commonly used options presented in this section, you should have no trouble using the other available options.

Note – If you do not understand a form or a particular field in a form, press F1 or select the HELP button at the bottom of the form. The system will provide an explanation.

Adding a New User

To add a new user to the system:

1. Log in as sysadm:

```
login: sysadm
Password:
```

2. Select the User menu. Your screen will look similar to this:

```

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

Note – An arrow (->) to the right of a menu option indicates that option leads to another menu.

3. Select the User Management option, then select Add a New User. The screen will look similar to this:

```
Disk      File      Machine  Software  User      Help      Quit
Add a new user to the system

                                Add a New User
-----
                                Current User Parameters
Full name:
Login ID:                               User ID: 668
Home directory:
Login group: <other >                   Group ID: 1
Login shell: </bin/sh >
Assign password? <YES>

[OK]      [CANCEL]      [HELP]
```

The Add a New User form displayed above contains these fields:

Full name:

Type the full name of the user who will own this account. For example, type Robin Hood. You cannot skip this field.

Login ID:

The login ID is the name by which the system will identify the user. Enter an appropriate login ID for the user, for example, robin.

User ID:

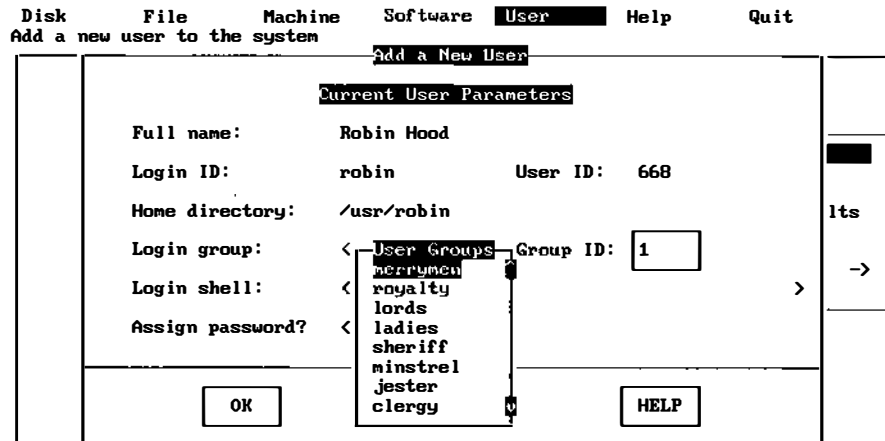
The User ID is the number used by the system to associate data files with a login ID. The system supplies a default user identification number (UID) in this field, automatically incrementing the last assigned number by one each time you add a new user to the system. Press Enter to accept the default UID. If you wish to assign a different user ID number, you can override the default number by typing a new number over it.

Home directory:

This is the directory in which the user is placed at login. The system supplies the path name for the user's home directory in this field. Press Enter to accept the default path name, or type in another path name if you do not want to use the default path name.

Login group:

This is the group to which the user is assigned. The system provides the name of the user's default login group in this field. To select a different login group, press the spacebar to pop up the list of available groups. Note that if you have not added any groups to your system, only other will be displayed. Your screen will look similar to this:



Move to the login group you want and press Enter to select it.

Group ID:

This information field contains the group identification number assigned by the system. You cannot alter this field.

Login shell:

This field contains the user's default login shell. To select a different shell, pop up the list of available shells. Your screen will look similar to this:

```

Disk      File      Machine  Software  User      Help      Quit
Add a new user to the system
Add a New User
Current User Parameters
Full name:      Robin Hood
Login ID:       robin          User ID: 668
Home directory: /usr/robin
Login group:    <merryman>      Group ID: 102
Login shell:    <
Assign password? <
User Shells
/bin/sh        Bourne shell
/bin/csh       C shell
/bin/ksh       Korn shell
/bin/rsh       restricted shell
other          custom shell
OK
  
```

Move to the shell you want and press Enter to select it.

Assign password?

This field allows you to assign a password for the new user. If you do not want to assign a password, type no. To assign a password, type yes. If you type yes, the following prompts will appear on your screen:

```

New password:
Type new password again:
  
```

Type in a password that is a unique word between 6 and 12 characters long. It may include upper- and lowercase characters, numbers, and symbols. It is a good idea to include at least two alphabetic characters and one number or symbol. Type the password again at the second prompt. (Note that if you do not assign passwords, users will have to set one the first time they log in.)

- Now that you have provided the system with the information necessary to add the user, the system creates the new account.

Adding a New Group

To add a new group, follow this procedure:

- From the User menu select Group Management, then select Add a New Group. Your screen will look similar to this:

The Add a New Group form displayed above contains these fields:

Group Name :

Type the name of the group in this field. It should be a word, number, or abbreviation that is between three and eight characters. A group name may only consist of lowercase letters and/or numbers. Uppercase letters and special symbols are not permitted.

Group ID:

The system supplies a default group identification number (GID) in this field. Press Enter to accept the default GID. If you want to

assign a different GID number, you can override the default by typing a new number over it. Note that since the numbers from 0 to 100 are reserved, `sysadm` will not accept a number below 100.

2. Now that you have provided the system with the information necessary to add the group, you can either install or cancel the entry. Select the `OK` button at the bottom of the form to install the entry. Select the `CANCEL` button to cancel the entry. You will be returned to the Group Management menu.

Managing Passwords

To log in to the system, both the login name and password must be used. Since the login names are known to other users on the system, the password is very important to system security. Observe the following guidelines when you use or assign passwords:

- A password should be hard to guess. Do not use names, birthdays, telephone numbers, or words that are of personal significance. For example, if you are known to be an enthusiastic sports fan, it is not a good idea to use the name of your favorite team.
- Passwords should be at least six characters long with at least two alphabetic characters and one non-alphabetic character. Note that if you are not the superuser, you must follow these guidelines in choosing a password.
- Never record your password or someone else's on paper.
- Make sure all system users understand the importance of keeping passwords secret.
- Encourage your users to change their passwords occasionally.

Aging User Passwords

Password aging allows the system administrator to set time requirements on passwords to make it harder for others to break into your system. After a specified period of time, passwords expire, and users are forced to enter new ones. Users are also prevented from changing a new password for a specified time.

When a login is first assigned, there is no aging. Password aging must be assigned by the system administrator (as the superuser) using the `passwd` command.

Password aging information consists of the following variables:

MINWEEKS = *number*

number defines how soon a user can change a password after it was last changed. **MINWEEKS** is defined for all users on the system in `/etc/default/passwd`. The system administrator can also change this variable for each individual user account using the `passwd` command. The default is 0.

MAXWEEKS = *number*

number defines the maximum amount of time a user can retain a password for a user account. **MAXWEEKS** is defined in `/etc/default/passwd`. The system administrator can also change this variable for each user account using the `passwd` command. The default is 1000.

IDLEWEEKS = *number*

number defines the length of time a user's password account is allowed to remain idle without being changed. **IDLEWEEKS** may be defined in `/etc/default/login`; by default, it is not defined.

The general form of the `passwd` command used to set password aging is:

```
# passwd -l -xmax -nmin login_name
```

The `-x` option specifies the maximum number of days (*max*) that a password is valid for the user named *login_name*. If *max* is 0, password aging is turned off for that login.

The `-n` option specifies the minimum number of days (*min*) that a password is valid for the name login. This means that *login_name* is forced to use the same password for at least *min* days.

For example, to force user `robin` to change his password every 30 days and to keep any assigned password for at least a week, type:

```
# passwd -l -x30 -n7 robin
```

See `passwd(1)` for a complete description of the `passwd` command.

Changing Other Users' Passwords

Ordinary users can use the `passwd` command to change their passwords any time (depending on how password aging is set). As superuser, you can change another user's password without knowing his or her password. This is useful when someone forgets a password for some reason.

To change a password, type:

```
# passwd login_name
```


When you first installed your system, you tailored several system parameters, such as the time zone and date. As you become more familiar with the INTERACTIVE UNIX Operating System, you may wish to customize your system to better suit your specific working environment.

An INTERACTIVE UNIX System is usually customized by modifying one or more *configuration files*. These are files that store information that affects the environment of an individual user or affects the system on a global basis. There are several configuration files on the system that can be modified to tailor the environment to meet system-wide or individual requirements.

Every INTERACTIVE UNIX System user operates in a unique environment that is controlled by the shell, the INTERACTIVE UNIX System command interpreter. A number of shell *variables* are defined at the time the user logs in to control many of the default actions of the user's login session. A variable is a word that is assigned a value and has a special meaning to the INTERACTIVE UNIX Operating System. For example, the default terminal type is a shell variable that is usually set for each user. Additional shell variables determine how commands issued by a user are run.

Many INTERACTIVE UNIX System configuration files are shell command scripts or contain shell variable assignments.

Each file has a default configuration that can often be tailored through the use of *comments*. A comment is a part of a file that is not processed by the program that reads the file.

- In a shell script, a comment is preceded by a #.

- In a C program (C is the name of the programming language used to write UNIX System programs), a comment is bracketed by /* and */.
- In an INTERACTIVE UNIX System text processing file, a comment is preceded by \" or '\ '.

Comments are ignored by the system. Thus, you may activate or inactivate commands in system configuration files by adding or removing the comment character(s). Many configuration files on the INTERACTIVE UNIX System use comment lines to explain the function of a command sequence.

Environment Variables

A number of shell or *environment variables* are set automatically by the system for all users. An environment variable is assigned the value of a *string* (a word, file name, command, or other sequence of characters or letters). A system administrator may want to change some of these variables in the system's default profile (/etc/profile), or an individual user may want to change them in his or her own .profile.

When using the Bourne (/bin/sh) or Korn (/bin/ksh) shell, the syntax for assigning a value to a variable is:

```
variablename=value
```

(Note that there are no spaces.) For example, the HOME variable is set equal to the path name of the user's home directory:

```
HOME=/usr/deb
```

The environment variables that are usually defined by default are listed alphabetically below.

CDPATH

Defines the paths to be searched for an argument to the cd command. By default, the current directory is searched.

HOME

Defines the path name of your login directory. The value of HOME is set when you log in and should not be changed. HOME is specified in `/etc/passwd`.

HZ

Defines the number of ticks per second for the system clock.

IFS

Defines the internal field separator characters. The shell initially sets these characters to include the space (blank), tab, and new-line characters.

LOGNAME

Defines your login name and is set when you log in to the system. This variable is often referred to by shell programs. LOGNAME is specified in `/etc/passwd`.

MAIL

Defines the full path name of the directory where you will receive mail from other users. MAIL is usually kept in `/usr/mail` (that is, `/usr/mail/LOGNAME`).

MAILCHECK

Defines the interval at which the system checks for new mail (in seconds).

PATH

Defines the directory search path for commands. By default, this variable includes the current directory, the `/bin` directory, and the `/usr/bin` directory.

PS1

Defines the primary shell prompt. By default, the shell prompt is set to `$` for regular INTERACTIVE UNIX System users and `#` for users who log in as root.

PS2

Defines the secondary shell prompt. By default, the secondary shell prompt is set to `>`. When this prompt appears, it means that additional information (input) is needed for the command to run.

TERM

Defines your terminal type for certain programs (such as screen editors).

TZ

Defines the time zone in which the computer is located.

The System Default Profile

If you use the Bourne shell (`/bin/sh`), when you first log in to the INTERACTIVE UNIX System, your working environment is defined by the default system profile located in `/etc/profile`. This file is a shell script that is executed each time the Bourne shell is started. It contains the commands needed to initialize your environment and the commands common to all users.

`/etc/profile` executes a sequence of commands and sets a number of shell (environment) variables. `/etc/profile` also runs a number of commands. It displays the message of the day and runs `umask`, the command that determines the default protections for new files and directories created by users. (Refer to Chapter 27, "Administering System Security," for more information about the `umask` command.) In addition, `/etc/profile` can be tailored to run other programs that should be run when a user first logs in, for example, the news program.

Here is a sample `/etc/profile` file:

```
#ident "(#)profile      2.9 - 93/04/05"

# This is the profile that all logins get before using their own
# .profile.

trap "" 1 2 3
umask 022                # set default file creation mask

# Only read TIMEZONE if TZ isn't already set (login usually sets it)
if [ -z "$TZ" ]; then
    . /etc/TIMEZONE
fi

case "$0" in
-sh | -rsh | -ksh)
```

```
# calculate available disk space in root filesystem.
echo "" # skip a line
[ -x /usr/bin/awk -a x"$LOGNAME" != x"sysadm" ] && /etc/dfspace

# issue message of the day
trap : 1 2 3
echo "" # skip a line
if [ -s /etc/motd ] ; then cat /etc/motd; fi
trap "" 1 2 3

# set default attributes for terminal
stty erase `^h` echoe

if [ x"$TERM" = x ]; then # if TERM isn't set, set to unknown.
TERM=unknown # unknown terminal type
export TERM
fi

# check mailbox and news bulletins
if mail -e ; then
echo "you have mail"
fi
if [ x"$LOGNAME" != x"root" -a -d /usr/news ]; then
news -n
fi
;;

-su)
:
;;
esac

export PATH;

trap 1 2 3
```

You must be logged in as root to modify the default system profile. Use an editor supplied with your system to modify this file.

The User's .profile

Individual users can further customize their environment by creating a personal version of `/etc/profile`, named `.profile`, which is stored in the user's login directory. `.profile` may be used to reset any of the environment

variables listed above to suit the individual's requirements better than the system defaults. The system reads `.profile` *after* it reads `/etc/profile`, so the variables set in `.profile` "override" the system defaults. Here is an example of a typical `.profile`:

```
PATH=/bin:/usr/bin:$HOME/bin
PS1="unix$ "
export PATH PS1
```

The `export` statement is used to make the variables apply to every environment the user enters.

After creating or making changes to the `.profile`, you can initiate the changes by logging off and logging back in again, or you can type:

```
$ . .profile
```

and press Enter. The shell will reinitialize your environment. The dot (`.`) is a special shell command used to execute commands in command scripts such as `.profile`.

To look at the variables set in your environment, you can use the `env` command. Type:

```
$ env
```

and press Enter.

You can set an environment variable temporarily, so that it applies only to the current login session, by typing it at the system prompt. For example, if you want to temporarily add the directory `/newdir` to your search path, type:

```
$ PATH=$PATH/newdir: export PATH
```

and press Enter. For that login session, your search path is set to your regular search path and also `/newdir`. The next time you log in, your `PATH` variable will be set from `/etc/profile` or `$HOME/.profile`, as usual.

Setting the System Date and Time

System time under the INTERACTIVE UNIX Operating System is controlled and affected by several different factors. Each of the following items plays a role in the date and time that users and programs observe:

- System clock
- CMOS RAM
- User environment
- Daylight Saving Time
- `setlocale` program

Your computer has an electronic clock powered by a small battery. This clock runs all the time, even when the computer is off. The computer operator can set the date and time that the clock maintains in one of four ways:

- The hardware manufacturer's *setup* program
- The ROM BIOS setup program, accessed by a "hot-key" before booting the system
- The `date` command
- The `sysadm setconf` command

The actual time displayed by the clock depends on the settings stored in the CMOS RAM (CoMplementary Oxide Silicon). This type of memory requires very little energy to maintain. The CMOS RAM memory is powered by the clock battery, even when the system is off. In addition to the date and time, system configuration information is also stored here. Usually, the manufacturer's *setup* utility or BIOS *setup* is used to change the system date and time.

If you use the `sysadm setconf` command or the `date` command to change the date or time, the change will be made permanently, updating both the time stored in the CMOS RAM and the UNIX System time. (See the *INTERACTIVE UNIX System Maintenance Guide* for a complete list of all the `sysadm` menu bypass commands.)

When the UNIX System kernel is booted, it reads the system date and time from the CMOS RAM. The UNIX System determines which time zone it is running in from the TZ environment variable (set by `/etc/init` as part of the system startup process). The TZ environment variable contains the main local time zone, the difference between the main local time zone and Greenwich

Mean Time (GMT), and a possible alternate time zone, for example, for areas that use Daylight Saving Time. The information in TZ is used in the calculation to convert the local time from the CMOS RAM to GMT, which is the time used internally by the UNIX System kernel. When the system displays dates and times, it converts GMT to the local time, again using the TZ environment variable to adjust for the local time zone. The initial time zone used by the system is set when the system is installed.

The user's environment affects programs that use or display the date and time, for example the `date` program. When the user logs in, the `login` program sets the TZ environment variable from the TZ value in the file `/etc/TIMEZONE`. The `/etc/init` program initially reads `/etc/TIMEZONE` and passes the TZ value specified therein to each login process. An individual `.profile` (Bourne or Korn shell) or `.login` (C shell) file should only need to change TZ if that user needs a different time zone from the rest of the system, for example, for a user in a branch office.

In areas that use Daylight Saving Time, local time is moved forward or backward at the appropriate times of the year. If the system is running during a change to or from Daylight Saving Time, the programs that use or display dates or times will correctly reflect the new date and time as long as the TZ environment variable appropriately specified the alternate time zone. However, the CMOS RAM time does not automatically change, so the next time the system is booted, the kernel will reinitialize the date and time based on the CMOS RAM, and it will thus revert to the old time. The date and time in the CMOS RAM must be updated manually using one of the methods specified above.

Resetting the System Date and Time

Use the `sysadm setconf` command to access the form that will allow you to reset the system date and time. Your screen will look similar to this:

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

Current 'sales' System Configuration

Machine name: sales      Date: Apr 14 1994      Time: 08:59:00 PM

Number of diskette drives: <2>      Reset time?      <no >
Drive 0 type: <3.5-inch, 1.44MB>      Reset date?      <no >
Drive 1 type: <5.25-inch, 1.2MB>      Time zone: <PST8PDT >
Math coprocessor present? <yes>      Daylight saving: <yes>

OK      CANCEL      HELP
```

This form displays a number of fields that you can explore on your own. Only the fields needed to set the date and time are discussed here.

1. Tab to the `Reset time?` field and select `yes`. The system displays another form in which you can use the spacebar to set the current hour, minute, second, and to select a.m. or p.m.
2. When you have completed the form, select the `OK` button at the bottom of the form. The system returns to the `Display/Change System Configuration` form.
3. Tab to the `Reset date?` field and select `yes`. The system displays another form. Use the spacebar to set the current month, day, and year, then select the `OK` button to return to the `Configuration` form.
4. If you want to change the time zone, Tab to the `Time zone:` field and use the spacebar to pop up a list of time zones. Select the time zone you want to use; you will be returned to the `Configuration` form.

5. Tab to the Daylight saving: field and use the spacebar to select yes or no.
6. Select the OK button at the bottom of the form. The system date and time are set.

Setting Up a Message of the Day

Your system may be set up so that a **message-of-the-day** is displayed whenever a user logs in to the system. The text of the message is stored in the file `/etc/motd`. You can modify this file, using an editor delivered with your system, to add important messages that should be seen by all system users. It is a good place to put messages regarding scheduled system downtime or other daily reminders. Here is a sample `/etc/motd` file:

```

*****  The system will be down tonight  *****
*****  from 5:00 p.m. to 7:00 p.m. for  *****
*****  scheduled system maintenance.    *****

*****  The new Networking Package is    *****
*****  installed. Problems to operations. *****

```

The message-of-the-day is displayed each time a user logs in to the system and whenever you use the `sysadm` command or `login`.

Changing the News

The message of the day should be used for important, but short announcements. If you have longer messages you want to put in a location accessible to all users, use the `/usr/news` files. Use one news file per news item. To add or change news, create a file in the news directory and type in the news you want system users to read. To read the news after logging in, each user must type `news` and press Enter. See `news(1)` for more information about news.

Automatic Program Execution

The cron Program

The cron program runs other programs automatically at specified times. This program and, more specifically, the crontab command, allow you to run programs during off hours such as:

- File system administration
- Long running, user-written shell procedures
- Cleanup procedures

Any task that needs to be done repeatedly at a specific time can be put into the cron file, which is located in the `/usr/spool/cron/crontabs` directory. If authorized by the system administrator, users can use the crontab command to establish their entries.

The crontab command has several options.

Command	Result
crontab	Copies the standard input from the screen into a directory that holds all users' crontabs.
crontab <i>file name</i>	Copies the named file into a directory that holds all users' crontabs.
crontab -r	Removes a user's crontab from the crontab directory.
crontab -l	Lists the crontab file for the invoking user.

See *crontab(1)* for additional information.

Each line in the crontab file defines one procedure. For example:

```
minute hour day month day-of-week command
```

Each field is defined shown here.

Field	Definition
<i>minute</i>	0-59
<i>hour</i>	0-23
<i>day</i>	1-31

Field	Definition
<i>month</i>	1-12
<i>day-of-week</i>	0-6 (0=Sunday)
<i>command</i>	The command to be executed at the time specified.

The following rules apply to the first five fields:

- Two numbers separated by a hyphen indicate a range of numbers between the two specified numbers.
- A list of numbers separated by commas indicates that only the numbers listed will be used.
- An asterisk specifies all legal values.

For example, `0 0 1,14 * 2` indicates a command will be run on the first and fourteenth of each month, as well as on every Tuesday. If a percent sign (%) is placed in the command field (sixth field), the INTERACTIVE UNIX System translates it as a new-line character. Only the first line of a command field (the character string up to the percent sign) is executed by the shell. Any other lines are made available to the command as standard input.

For example, suppose a file called `anyfile` contains the following cron entry:

```
0 0 1 * * mailx $LOGNAME % Subject: Call Mom! % now
```

When the command line `crontab anyfile` is executed, the user whose login is `$LOGNAME` will get a reminder mail message with `Call Mom!` as the subject on the first of every month.

Automatic System Cleanup

The INTERACTIVE UNIX System should be cleaned up occasionally. Fortunately, the `crontab` command and `crontab` file make this task easier. You can specify cleanup jobs (for example, remove aged files) and the time at which you want them to execute in the `crontab` file.

Your system comes with some default cleanup procedures already defined. These cleanup procedures are done by the `root` login under the control of `crontab` each Sunday morning at 5:17 a.m. The file `/etc/cleanup` defines what cleanup procedures are done.

Two of the files cleaned up automatically each Sunday morning are:

`/etc/wtmp`

This file contains a history of system logins. Every time a user logs in, a record is made in this file. If not cleaned up, the size of this file will grow forever. Instead of deleting the contents of this file yourself, let cron do it for you.

`/usr/adm/sulog`

This file contains a history of users that use the `su` command to switch logins. As a security measure, this file should not be readable by other users. See `su(1)` for additional information.

Log in as `root` and execute `crontab -l` to see the crontab entry that executes `/etc/cleanup` as well as other cleanup routines for UUCP (Basic Networking). Examine `/etc/cleanup` to see the default routines run each Sunday morning. To change the way cleanup jobs are performed, edit `/etc/cleanup` and modify the root crontab using the `crontab -l` and `crontab` commands.



Caution – Never edit `/usr/spool/cron/crontabs` directly.

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 (such as `mail` and `mailx`), which then sends it to `sendmail` for delivery.

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 the way 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 the message to another machine.
- It sends the message 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.

Additionally, each user may have a `$HOME/.forward` file which `sendmail` checks for routing instructions.

`smail` is a 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`.

Sample Mail System Configurations

```
mail    →    uucp
```

This is the default USL mail configuration.

```
mailx   →    sendmail   →    uucp
```

This is a UUCP-only installation where the user is not using `smail`.

```
mailx   →    sendmail   →    tcp
```

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.

```
mailx   →    sendmail   →    uucp
                               →    tcp
```

This is a network and UUCP configuration where the user is not using `smail`.

```
mailx    →    sendmail    →    smail    →    uucp
                                     →    tcp
```

This is a network and UUCP configuration where the user is using smail.

The Location of Mail-Related Files

Table 21-1 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.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

sendmail Installation Instructions

Before configuring your system to use sendmail, you should have a basic understanding of mail system components and the skills required of a system administrator. In addition, 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 the 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 books *Managing UUCP and Usenet* and *sendmail*, published by O'Reilly and Associates, Inc. They can be contacted at 1-800-998-9938 or 1-707-829-0515.

Configuring the Mail System Using `sysadm`

For information about installing optional subsets, such as *Basic Networking Utilities* and *sendmail*, refer to the *INTERACTIVE UNIX System Installation Guide*.

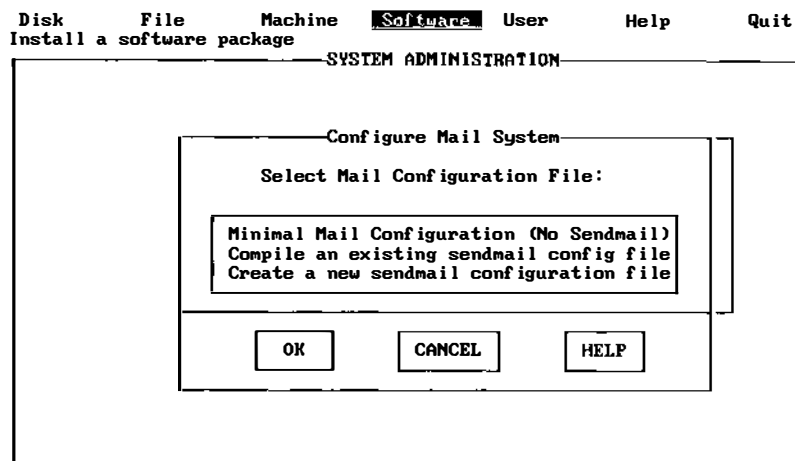
After using `sysadm` to load the necessary software packages on your system, use the `sysadm` program to configure your mail software. `sysadm` allows you to select one of four template files that sets up the default values for the method of mail delivery. This file becomes your `sendmail.cf` configuration file after you tailor the values for your system. Defaults are set only for components of your particular mail system.

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:



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:

Note – On some systems, the system automatically fills in the domain name and host name for your system.

```

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

```

3. In the first field, press the spacebar 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: < [ ] >
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

```

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 snail? <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. The `com` suffix is used by commercial companies to distinguish them from educational institutions (`edu`), the government (`gov`), or military (`mil`) sites. For 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 for more information.



Caution – 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 that you have chosen, 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. Tab to the next field you want to change and press the spacebar to cycle through your choices. Press 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:

```

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: 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
                       127.213.32.2
Use relays?
Use smail?

          [OK]          [CANCEL]          [HELP]
    
```

If you already have a valid `/etc/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 the `sendmail.cf` file 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
Use mailhu
Use namese
Use relays
Use smail?

Current Mail Relay Information

Internet:
UUCP:
CSNet:
Bitnet:

.32.2

[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 all possible mail systems and may need further tuning.

7. Type in the appropriate information for your system, then press Enter to move from field to field. When you have finished entering the information for your system, move to the OK button and press 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?
Use nameserver?
Use relays?
Use smail? <yes>      Smrt Hst:
          8.212.32.2
          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 automatically “freezes” (compiles) 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.

Understanding INTERACTIVE UNIX System File Systems

22 

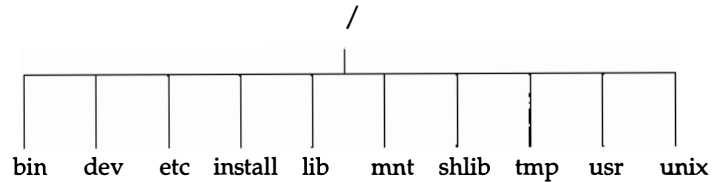
Many of the system administrator's responsibilities involve maintaining and monitoring file systems. This chapter explains how a file system is structured and the naming conventions that are used to access file systems and peripheral devices (usually hardware). You will use the `sysadm File` menu to perform most file system maintenance. However, before you can begin using the *utilities* available with the `sysadm` command, you must understand the definition and the structure of an INTERACTIVE UNIX Operating System file system.

What Is an INTERACTIVE UNIX System File System?

An INTERACTIVE UNIX System *file system* is a complete directory structure that is contained on a diskette or a fixed disk. It contains a collection of files arranged in a hierarchical order. On a diskette, a file system usually takes up the entire volume. On a fixed disk, a file system is associated with a single subpartition of the INTERACTIVE UNIX System partition on the fixed disk.

There is at least one permanent file system established on the first subpartition of the INTERACTIVE UNIX System partition on the first fixed disk. This permanent file system is known as the *root file system*. The main (top level) directory in the *root file system* is called the *root directory*. It contains all of the important files and subdirectories that are required to run the INTERACTIVE

UNIX Operating System. The root directory is automatically created and given the unique name of / (slash) when your system is installed and initialized. A typical root file system looks like this:

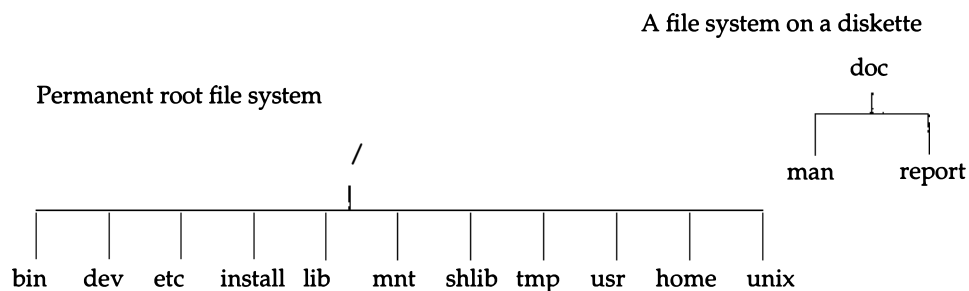


In this example, the file `unix` in the root directory contains the kernel. The remaining entries are files and subdirectories that contain text files, command files, shell scripts, and other directories that are used by the operating system.

In addition to the permanent root file system, you can establish additional file systems on fixed disks or diskettes. Additional file systems can be *mounted* on the original root file system. Mounting a file system attaches it to the root directory and makes it accessible to users. If the file system is located on a portion of the fixed disk, it can be automatically mounted when the system is brought up. It is often useful to establish multiple file systems on the fixed disk to improve system performance or to prevent unauthorized access to sensitive data. Fixed disk file systems are created when you install your INTERACTIVE UNIX System and when you install additional fixed disks on your computer. If the file system is established on a diskette, it must be mounted before it can be accessed.

The figure below illustrates a typical INTERACTIVE UNIX System file system structure. The root file system contains one additional mounted file system, `home`. In addition, a user owns a diskette that contains a file system labeled

doc. It has not been mounted. To be made accessible to users, the diskette must be physically present in the diskette drive and the file system on it must be mounted.



All files and directories in a file system are accessed using a path name. Path names have the following form:

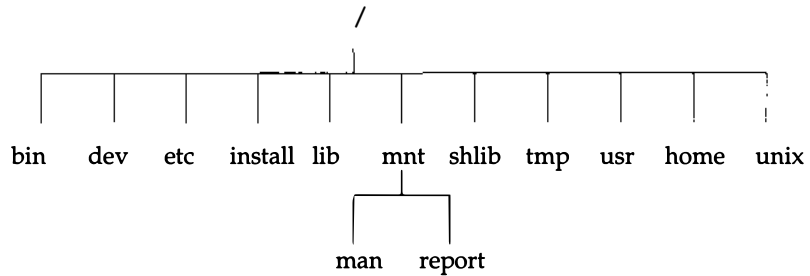
/name1/name2/name3

where *name1* and *name2* are directory names and *name3* (the last name in the path name) is either a directory name or a file name.

The first slash (/) in the path name is the name of the `root` directory. The subsequent slashes in the path name are used to separate the directory names and file names. Once you have mounted a file system, it appears to be an ordinary directory that is part of the `root` file system and is accessible by

standard INTERACTIVE UNIX System naming conventions. For example, when the `doc` file system described above is mounted under the `/mnt` directory, the file system will look like this:

Permanent root file system



To access the file called `report`, use the path name:

```
/mnt/report
```

The file system is still the one labeled `doc` on the diskette, but the label is not used while the file system is mounted on the `root` directory.

File System Naming Conventions

Before you can begin any routine maintenance of the file system, you must understand the basic conventions that are used to name file systems and devices. When you mount a file system, back up a file system, or check a file system for errors, you may need to identify the device on which the file system is located. This section outlines the naming conventions for file systems on the INTERACTIVE UNIX System. The next section outlines the naming conventions for the most common devices found on your system. For more detailed information, refer to the *INTERACTIVE UNIX System Maintenance Guide* and the *INTERACTIVE UNIX System Installation Guide*.

A file system name or “label” may be up to six characters in length and may contain upper- or lowercase letters and/or numbers. You should not use any characters in a file system name that have a special meaning to the shell. For example, *, ?, and & are used by the shell and should not be used in file system labels.

It is a good idea to give the file systems meaningful names. For example, if you are creating a new file system that will be used exclusively by the documentation department, you might label the file system `doc`. Depending on the size of your fixed disk and the INTERACTIVE UNIX System partition, the INTERACTIVE UNIX System creates one or more file systems when you install it. If only one is created, it is named `root`; if two are created, they are named `root` and `usr`. If created, the third file system is named `home`. Subsequent file systems are named `home2` through `home4`. If you wish, these names can be changed during installation.

Mount Point Conventions

A file system is attached to the `root` directory through a process called mounting. Mounting a file system “tells” the system where the file system is located. The file system is mounted onto a directory in the `root` directory with a system administration procedure called `mount` (this is discussed in more detail in Chapter 24, “File System Maintenance”).

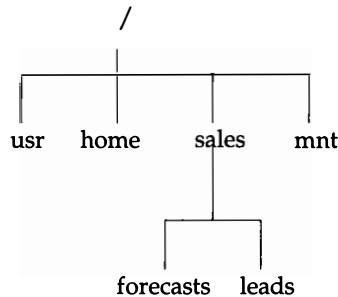
You must create a directory (using `mkdir`) to be used as the *mount point* for the file system. A mount point is a directory where the file system will be installed. Only one file system at a time can be installed on a mount point directory. The mount point directory may already contain other files and directories, but they will be inaccessible if a new file system is mounted on that directory. These existing files and directories are not deleted; they remain “hidden” until the newly mounted file system is deactivated or “unmounted.” Users should be warned not to put any files in the mount point directory because the files will be unavailable while a file system is mounted there.

Both fixed disk and diskette file systems must be mounted to be used. By convention, diskettes are often mounted under a directory named `/mnt`. This is a convenient way to remember the file system name of all diskettes. It also eliminates the need to create extra directories under the `root` directory. By convention, directories created under the `root` directory for mounting fixed disk file systems are usually given the same name as the file system itself.

The diagram below illustrates how a directory serves as a place to mount a file system. The root file system is shown below on the left, and the unmounted sales file system on the fixed disk is shown on the right:



After you have created a directory under the root directory called sales and have mounted the sales file system there, it would look like this:



The complete path name to the directory where the sales file system is mounted is /sales.

You can create several file systems on a fixed disk during system installation or when adding a new fixed disk to your system. Refer to the *INTERACTIVE UNIX System Maintenance Guide* and the *INTERACTIVE UNIX System Installation Guide* for more information about creating file systems. Create a new directory under the root directory for each new file system. The directory name should be the same as the file system label. This makes it easier to remember the location of each new file system. For example, if you create a new file system labeled doc, create a new directory in the root directory

called `doc`. The complete path name to the directory where the `doc` file system is mounted is `/doc`. You can also use this convention for diskette file systems, but it is more common to use only one directory, `/mnt`, for mounting file systems on diskettes. (See the illustrations in “What Is an INTERACTIVE UNIX System File System?” at the beginning of this chapter.)

Device Naming Conventions

The hardware devices attached to the INTERACTIVE UNIX System are named and accessed using ordinary path names. Device files are called *special files* because they do not contain data; they are references to the actual programs that run the peripheral devices attached to your computer. These programs are called *device drivers* and are used to control terminals, fixed disks, and diskette drives (among other things).

Note – Device names are arbitrary in most cases. The INTERACTIVE UNIX System is supplied with certain pre-established device names. These are adequate for most needs. System administrators may create additional special files or links between existing special files at their discretion. (Refer to *ln(1)* for more information about linking files.)

By convention, device files reside in a special directory called `/dev`. There are two basic types of device files: *character files* and *block files*. Character devices, such as terminals, generally process data one character at a time. Block devices, such as fixed disks, access hardware one block (sector or record) at a time. The INTERACTIVE UNIX System accesses a file system via a block device.

The device files for both fixed disks and diskettes reside in two parallel subdirectories, `rdsk` and `dsk`, under the directory `/dev`. `rdsk` stores character device files, and `dsk` stores block device files. Each device file has a unique name, which may appear in the `dsk` directory or the `rdsk` directory, or both. The device file name describes the physical attributes of the device it refers to. The naming conventions for diskettes and fixed disks are different; they are described in the following sections. As a system administrator, you need to know the names of the devices on your system because certain errors are reported in connection with specific devices. Device names for mountable file systems are always found in `/dev/dsk`, unless they are XENIX file system names, which are located in `/dev`.

Note that some special files are also located in `/dev/SA` and `/dev/rSA`. These directories are used by the `sysadm` program to determine some device names.

Partitions and INTERACTIVE UNIX System File Systems

When the INTERACTIVE UNIX System is installed, its `fdisk` program is used to divide the fixed disk into *partitions*. You may have only one partition that takes up the entire fixed disk, or you may have several partitions, each containing its own operating system, such as UNIX or DOS. The partitions that divide up the entire fixed disk and are used to hold different operating systems are referred to in this document as `fdisk` partitions.

Even if you have multiple `fdisk` partitions, the INTERACTIVE UNIX System is installed on only one. This partition is itself subdivided into a number of sections that are sometimes referred to in technical documents as partitions. To avoid confusion, we refer to these subdivisions of the INTERACTIVE UNIX System partitions as subpartitions, since they are divisions of the `fdisk` partition on which the INTERACTIVE UNIX Operating System is installed. Both types of partitions are described in the following section.

Device Naming Conventions for Partitions on Fixed Disks

A fixed disk device name associated with an INTERACTIVE UNIX System partition has the following format:

```
ccontroller_numberdisk_numberspartition_number
```

The *controller*, *disk*, and *partition* numbers are 0-based (counting begins with 0 instead of 1).

A fixed disk device name associated with an `fdisk` partition has the same format, except that the partition number is preceded by a `p` rather than an `s`:

```
ccontroller_numbertarget_number[logical_unit_number]ppartition_number
```

controller_number

This number refers to the single controller that most systems have. The controller is named `c0`. If you add a second controller, it is named `c1`.

target_number

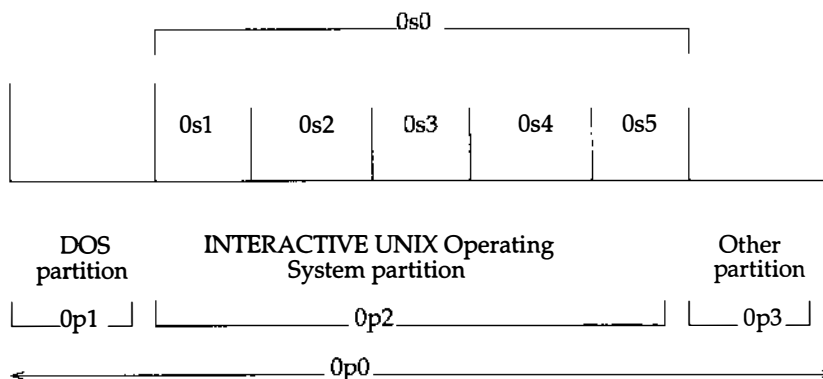
This number refers to your fixed disk. Your first fixed disk is usually named `t0`; if you have a second fixed disk, it is usually named `t1`. Each fixed disk's target number corresponds to its SCSI ID (for SCSI fixed disks), or its position on the controller (non-SCSI).

logical_unit_number

The *logical_unit_number*, or the option `l`, is used if there are multiple logical unit numbers on a single target. Generally this option is not needed.

partition_number

This refers to the `fdisk` partitions on your fixed disk (if preceded by a `p`) or to the "subpartitions" of your INTERACTIVE UNIX System partition (if preceded by an `s`). You must always specify the partition number when naming a device. A typical disk might be partitioned as in the following illustration:



`s0` refers to the entire INTERACTIVE UNIX System partition and is generally used for maintenance purposes (for example, in repairing a damaged disk). The first actual subpartition is called `s1`. Each subsequent partition number (`s2`, `s3`...) is incremented by one in *hexadecimal* notation. The tenth and last subpartition is `sa`, "a" being 10 in hexadecimal.

Certain INTERACTIVE UNIX System subpartitions are typically reserved for use by the system and cannot be accessed by user programs. These are: `s2` (the *swap area*, which is used to store portions of programs or data that have been

temporarily swapped out of main memory), `s8` (a small partition that stores the bootstrap program), and `s9` (used to store alternate sectors that will be substituted for bad sectors in other partitions).

The remaining subpartitions (`s1`, `s3` through `s7`, and `sa`) are used to store program and data files. Of these subpartitions, only the first, `s1`, is required, and it must be located on the first (primary) fixed disk. This subpartition contains the `root` file system, where system files and programs are stored.

If present, subpartition `s3` contains the `usr` file system, which is used to store some system files, and possibly users' files. Subpartitions `s4` through `s7`, if present, contain user-specified file systems, typically named `home` and `home2` through `home4`. These file systems are used to store users' files. Finally, partition `sa`, if present, contains the `tmp` file system, which is used to store temporary files and data. (If separate `usr` and `tmp` file systems are not created, their function is performed by the `usr` and `tmp` directories in the `root` file system.)

If you have a small fixed disk, it may contain only one partition, with the `root` file system located on it. If you have a larger disk, or have two fixed disks, then these will contain additional partitions. Subject to size constraints, you may specify during installation how your fixed disk is to be partitioned. Refer to the *INTERACTIVE UNIX System Maintenance Guide* and the *INTERACTIVE UNIX System Installation Guide* for more information about partitioning your fixed disk and optimal division of file systems between different fixed disks.

Generally, file system names should be unique. If file system names are the same, they cannot be mounted at the same mount point simultaneously. For example, if there is a file system named `var` on both the primary disk and the second disk, it is not possible to mount them both on `/` at the same time. However, file systems with the same name can be mounted at the same time if they are mounted on different mount points, such as mounting one `var` partition on `/tmp` and the other on `/usr/tmp`.

The default organization of INTERACTIVE UNIX System subpartitions is described in Table 22-1. The last two columns indicate, for both the first (primary) and additional disks, whether the specified partition is required or optional.

Table 22-1 Default Organization of Partitions

Partition Name	Contents	Primary Disk	Additional Disks
p0	Used for maintenance purposes only (refers to entire disk)	Required	Required
s0	Used for maintenance purposes only (refers to entire INTERACTIVE UNIX System subpartition)	Required	Required
p1	First fdisk partition	Required	Optional
p2	Second fdisk partition	Optional	Optional
pn	Additional fdisk partitions	Optional	Optional
s1	Contains the root file system	Required	No
s2	Contains the swap area	Required	Optional
s3	Contains the usr file system	Optional	Optional
s4	Contains user-specified file system	Optional	Optional
s5	Contains user-specified file system	Optional	Optional
s6	Contains user-specified file system	Optional	Optional
s7	Contains user-specified file system	Optional	Optional
s8	Contains the bootstrap area	Required	Required
s9	Contains alternate sector space	Required	Required
sa	Contains the tmp file system	Optional	Optional

These partitions are automatically created by the system when you install your disks.

Examples of Device File Names for Fixed Disks

The complete device file name for the second (first usable) partition of the first fixed disk on a system with one controller is `/dev/dsk/c0t0s1`.

Table 22-2 shows some common device file names for fixed disks that reside in the directory `/dev/dsk`. Your system may not require all of these special files. The controller, fixed disk, and subpartition numbers in the table are one-based.

Table 22-2 Common Fixed Disk Device File Names

Controller Number	Fixed Disk Number	Subpartition Number	Device Name
1	1	1	c0t1s1
1	1	2	c0t1s2
1	2	1	c0t2s1
1	2	4	c0t1s4
2	1	1	c1t0s1
2	1	2	c1t0s2
2	2	1	c1t1s1

Device Naming Conventions for Diskettes

The naming conventions for diskette devices are as follows:

```
fdiskette_number[ d, q ]number_sectors[ d, dt ]
```

The term *diskette_number* refers to your diskette drive number. 0 designates the first (or only) drive, sometimes referred to as the A drive. If you have two drives, the second drive is designated 1. Do not type the brackets shown in the statement above. They indicate the options that may be available to you. Use *d* if you have a double-sided, double-density drive, and *q* if you have a high-density (quad) diskette drive.

The term *number_sectors* refers to the number of sectors per track on your diskette. Use 8 for single-density diskettes, 9 for most double-density diskettes, 15 for high-density 5.25-inch diskettes, and 18 for high-density 3.5-inch diskettes. Select *dt* if you plan to use the entire diskette, including the first cylinder (the boot block). Select *d* if you plan to begin reading and writing on the diskette with the second cylinder.

Examples of Device File Names for Diskettes

If you have a high-density 5.25-inch (1.2 MB) diskette in the first drive and you plan to use the entire diskette, you will use the device file name `f0q15dt`. The complete path name to this device is `/dev/dsk/f0q15dt`.

If you have a high-density 3.5-inch (1.44 MB) diskette in the second drive and you plan to use the entire diskette, use the device file name `f1q18dt` to access the second drive. The complete path name to this device is `/dev/dsk/f1q18dt`.

Table 22-3 shows some standard diskette device file names. The most common device file name is listed first in each category, followed by other valid names, or aliases.

Table 22-3 Common Diskette Device File Names

File Name	Description
<code>/dev/dsk/f0d9dt</code> <code>/dev/dsk/f05d9t</code> <code>/dev/SA/disk0_360k</code>	First drive, 5.25" double-density diskettes, 360K
<code>/dev/dsk/f0q15dt</code> <code>/dev/dsk/f05ht</code> <code>/dev/SA/disk0_1.2M</code>	First drive, 5.25" high-density diskettes, 1.2 MB
<code>/dev/dsk/f0q9dt</code> <code>/dev/dsk/f03dt</code> <code>/dev/SA/disk0_720K</code>	First drive, 3.5" double-density diskettes, 720K
<code>/dev/dsk/f0q18dt</code> <code>/dev/dsk/f05qt</code> <code>/dev/SA/disk0_1.44M</code>	First drive, 3.5" high-density diskettes, 1.44 MB
<code>/dev/dsk/f0q36dt</code> <code>/dev/SA/disk0_2.88M</code>	First drive, 3.5" extra-density diskettes, 2.88 MB
<code>/dev/dsk/f1d9dt</code> <code>/dev/dsk/f15d9t</code> <code>/dev/SA/disk1_360K</code>	Second drive, 5.25" double-density diskettes, 360K
<code>/dev/dsk/f1q15dt</code> <code>/devdsk/f15ht</code> <code>/dev/SA/disk1_1.2M</code>	Second drive, 5.25" high-density diskettes, 1.2 MB

Table 22-3 Common Diskette Device File Names (Continued)

File Name	Description
/dev/dsk/f1q9dt /dev/dsk/f13dt /dev/SA/disk1_720k	Second drive, 3.5" double-density diskettes, 720K
/dev/dsk/f1q18dt /dev/dsk/f15qt /dev/SA/disk1_1.44M	Second drive, 3.5" high-density diskettes, 1.44 MB
/dev/dsk/f1q36dt /dev/SA/disk1_2.88M	Second drive, 3.5" extra-density diskettes, 2.88 MB

There are also four generic diskette device files that automatically determine the density of the diskette. These support 360K and 1.2 MB 5.25-inch diskettes and 720K and 1.44 MB 3.5-inch diskettes (the entire diskette).

/dev/fd0	Block device, first drive
/dev/fd1	Block device, second drive
/dev/rfd0	Character device, first drive
/dev/rfd1	Character device, second drive

Note that an unformatted 360K 5.25-inch diskette cannot be formatted in a 1.2 MB drive, and an unformatted 720K 3.5-inch diskette cannot be formatted in a 1.44 MB drive using the generic device files.

For ease of use, link your diskette special file to an easily remembered name. For example, to link a primary 3.5-inch 1.44 MB diskette drive to /dev/floppy:

```
# ln /dev/rdisk/f0q18dt /dev/floppy
```

Device Naming Conventions for SCSI Tapes

The naming conventions for tape devices are as follows:

```
[n] rmtcontroller_number target_number [logical_unit_number]
```

The n option should be used for devices that are "non-rewind."

The *controller_number* refers to the single controller found on most system. This is the boot controller or 0. If you have a second controller, it is named 1.

The *target_number* refers to your tape device. This is the SCSI identification number set by the manufacturer of the device.

The *logical_unit_number* is used if there are multiple logical unit numbers on a single target. Generally this option is not needed.

Table 22-4 shows some common tape device file names.


Table 22-4 Common SCSI Tape Device File Names

Controller Number	Tape ID Number	Device Name
0	0	[n]rmt0t0
0	1	[n]rmt0t1
1	0	[n]rmt1t0
1	1	[n]rmt1t1
2	0	[n]rmt2t0
2	1	[n]rmt2t1

Note that the first tape device on your system can be accessed using the name `/dev/tape` (or `/dev/ntape` for a non-rewind device). If you have a second SCSI tape device, you may find it convenient to create a link to it with a command like the following:

```
$ ln /dev/rmt0t1 tape2
```


Using Utilities With the Diskette Drive

23 

Utilities are provided with the INTERACTIVE UNIX System to read and write data to and from diskettes. Following are some of the basic diskette operations performed using these utilities:

- Formatting diskettes
- Copying files from the fixed disk to diskettes
- Copying files from diskettes to the fixed disk
- Duplicating diskettes

The INTERACTIVE UNIX System provides access to several types of diskettes. A diskette is accessed through either block or character (raw) device names. For information about INTERACTIVE UNIX System device names, refer to Chapter 22, “Understanding INTERACTIVE UNIX System File Systems.”

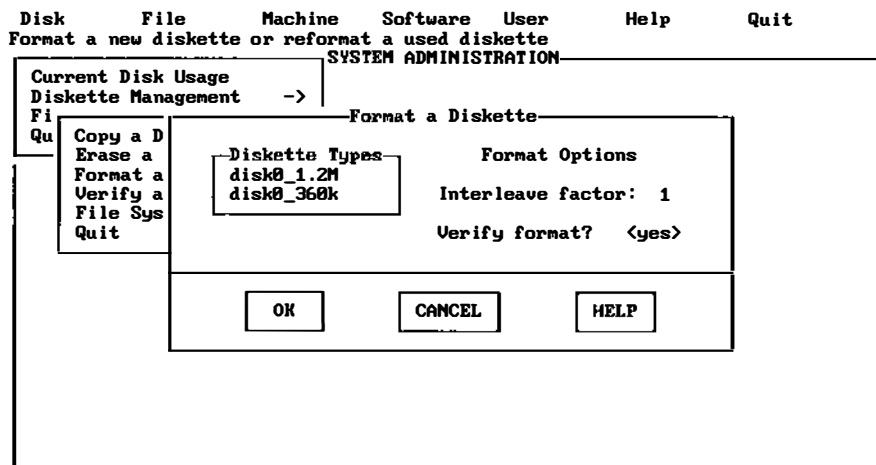
Formatting Diskettes

Before a diskette can be used in any diskette operation, it must be formatted. You can use the `sysadm` program to format diskettes, or you can use the `format` command directly.

Formatting a Diskette Using `sysadm`

To use `sysadm` to format a diskette, follow these steps:

1. If you are using a new diskette, access the `sysadm` Disk menu, select Diskette Management, then select Format a Diskette. Your screen will look similar to this:



2. In the first field, select the option that corresponds to the type of diskette in the drive you want to use (remember that `disk0` is drive A, and `disk1`, if available, is drive B). Check your manufacturer's documentation to determine this if you are unsure. Note that if you have a high-density diskette drive, you can format both low- and high-density diskettes. The first option selects a high-density diskette in the first (5.25-inch) diskette drive; the second option selects a low-density diskette in the first (5.25-inch) diskette drive. (Note that other options may appear on your screen depending on what type and how many diskette drives you have installed on your system.)
3. In the next field, select the interleave factor. The default interleave factor is 1, but you can specify a different interleave factor by typing over the default.
4. Specify whether you want the system to verify the format. By default the system will verify the format, but you can turn off verification by selecting the no option from the list in this field.

5. Select the OK button at the bottom of the form, then insert the diskette in the drive. The system formats your diskette. When formatting is complete, you will be asked whether you want to format another diskette. If you want to format another diskette, select the YES button at the bottom of the form; otherwise, select NO.

The Diskette Management menu includes a number of other options for working with your diskettes. After going through the Format a Diskette option, it should be easy to use the other options, following the on-line instructions. Remember that help is always available in the `sysadm` program when you press F1 or select the HELP button at the bottom of a form.

Formatting a Diskette From the Command Line

To format a diskette from the command line:

1. Determine the diskette type.
2. Insert the diskette into the drive.
3. Enter the `format` command followed by the appropriate character device name for that type of diskette.

The following command will format a 1.2 MB double-sided, high-density diskette:

```
$ format /dev/rdisk/f0q15dt
```

Copying Diskettes

If you want to make copies of a diskette, access the `sysadm` Disk menu, select Diskette Management, then select Copy a Diskette. The `sysadm` program will ask you to specify which diskette drive to use.

Note – If your system has more than one diskette drive of the same type (for example, two 1.44 MB diskette drives), you can make a copy directly from one drive to another. This is faster if you only have to make one copy.

The `sysadm` program will first ask you to insert the diskette, the one you want to make copies of, in the `From:` drive. An image of the diskette will be temporarily stored in `/usr/tmp` (unless the drive-to-drive option was selected). Next, you will be asked to insert a formatted diskette (see the previous section on how to format diskettes) in the `To:` drive. After the diskette has been copied, you can insert additional diskettes to make more copies.

Copying Files To and From a Diskette

Several methods can be used to copy files to a diskette. One method is to format a diskette, create a file system on the disk, mount the file system, and access the diskette as part of the INTERACTIVE UNIX System directory structure. This might seem a bit complicated, but it is a very convenient way to create a portable file system.

Simpler methods involve using the `cpio` command with other commands, such as `find` or `ls`, or using the `tar` command.

Copying Files Using cpio

The `cpio` command copies files to and from diskettes by taking a list of files from standard input and creating a file archive that can be redirected to a diskette. Several commands can be used to supply the file names to be archived by using the INTERACTIVE UNIX System shell pipe facilities. For example, to copy all the files in the current directory to a 1.2 MB diskette:

1. Insert a formatted 1.2 MB diskette into the drive.
2. Change directories to the directory containing the files to be copied.
3. Type:

```
$ ls | cpio -oc > /dev/dsk/f0q15dt
```

In this example, `ls` lists the files in the current directory and pipes the list to `cpio`. Then `cpio` archives them and redirects the output to the diskette drive.

Similarly, any files copied onto a diskette using the `cpio` command can be copied back to the fixed disk using different `cpio` options. The following example copies the files that were copied to a diskette in the previous `cpio` example back to the fixed disk into the current directory.

1. Insert the diskette containing the `cpio` archive into the drive.
2. Type:

```
$ cpio -ic < /dev/dsk/f0q15dt
```

Refer to `cpio(1)` for a complete description of the `cpio` command.

Copying Files Using the tar Command

As an alternative to `cpio`, you can use the `tar` command to transfer files to and from diskettes. The following example copies files from the fixed disk onto a 1.2 MB diskette:

1. Insert a formatted 1.2 MB diskette into the drive.
2. Change directories to the directory containing the files to be copied.
3. Type:

```
# tar -cf /dev/dsk/f0q15dt *
```

These files can then be copied back to the fixed disk using the `x` option to `tar`, as follows:

1. Insert the diskette containing the `tar` archive into the drive.
2. Change directories to the destination directory where you want to copy the files.
3. Type:

```
$ tar -xf /dev/dsk/f0q15dt
```

Refer to `tar(1)` for a complete description of the `tar` command.

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, you may also have chosen to create one or more user file systems, typically named /usr, /home, /home2, and so on. Refer to the *INTERACTIVE UNIX System Maintenance Guide* and the *INTERACTIVE UNIX System Installation Guide* for more information on manipulating file systems on fixed disks.

If you plan to store data on diskettes or to add a second fixed disk to your system, you will need to create and initialize additional file systems. You will use the `sysadm Disk` menu to facilitate file system management on your system. In this chapter you will learn about:

- Creating a file system
- Mounting a file system
- Unmounting a file system
- Checking the amount of disk space available for a file system
- Checking and repairing a file system

Creating a File System on a Diskette

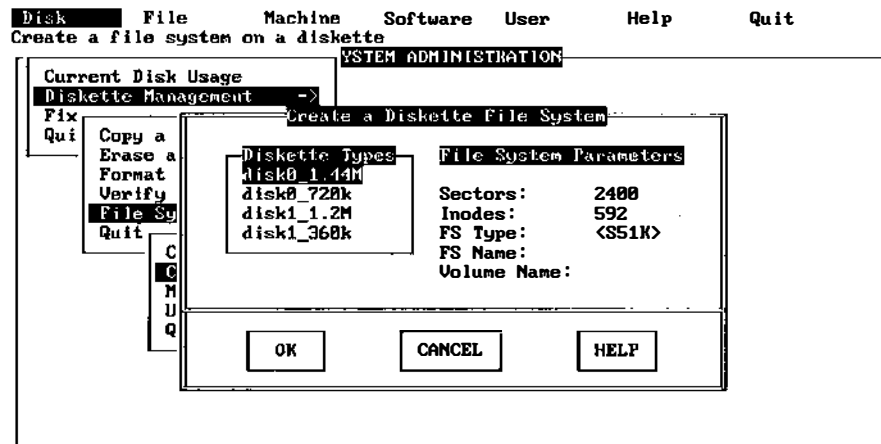
Before you can create a new file system on a diskette, you must format the diskette. To create a file system on a formatted diskette (a removable file system), use the `sysadm` command to access the `Disk` menu. Your file system will use the entire diskette.



Caution – If you plan to use the diskette solely to store data using the `sysadm` backup utilities, you do not need to create a file system on it. Any files or file systems found on the diskette are automatically destroyed by the backup procedures.

To create a file system on a formatted diskette, follow these steps:

1. From the Disk menu, select Diskette Management. Then from the File System Management menu, select Create a Diskette File System. Your screen will look similar to this:



2. In the first field, select the option that corresponds to the type of diskette in the drive you want to use (remember that `disk0` is drive A, and `disk1`, if available, is drive B). Check your manufacturer's documentation to determine this if you are unsure. The first two options select a high- or low-density 3.5-inch diskette in the first diskette drive; the third and fourth options select a high- or low-density 5.25-inch diskette in the second diskette drive.

Note – Other options may appear on your screen depending on what type and how many diskette drives you have installed on your system.

3. The next two fields indicate the number of sectors on the entire diskette and the number of inodes available. The system provides reasonable defaults based on the type of diskette you have chosen. Select the default values to accept them.

If you want to change the defaults, you can type new values over them. For more information about changing a default, press F1 in the appropriate field.

4. Press the spacebar in the FS Type: field to choose a file system type for the diskette you are creating. S51K indicates a System V 1K file system. S5L indicates a System V long name file system.
5. Type the file system name of the diskette file system at the FS Name: prompt. The file system name can be six characters long.
6. Type the volume name of the diskette file system at the Volume Name: prompt. The volume name can be six characters long.
7. Select the OK button at the bottom of the form when you have finished.
8. When the file system has been created, you will be asked whether you want to make another file system. If you do, select the YES button and follow the procedures given above. If you do not want to make another file system, select the NO button.
9. To access the files in the file system you have just created, refer to “Mounting a Diskette-Based File System” in this chapter.

Mounting a File System

There can be many file systems on an INTERACTIVE UNIX System. The file systems may be located on a number of devices, such as diskettes and fixed disk partitions. To use a file system, the INTERACTIVE UNIX System kernel must have information about the device where the file system is located and must know how to access the device. In addition, users must be able to access the file system with an INTERACTIVE UNIX System path name.

A file system cannot be used unless it is mounted on another file system, usually the root file system. File systems are mounted under the root (/) directory with the mount command or through the sysadm Disk menu. To mount a diskette-based file system, access the File System Management menu of the Diskette Management menu under Disk and select Mount a Diskette File System. Fixed disk file systems are mounted automatically each time you boot the system and are unmounted when you run the shutdown command. You may also use the sysadm option, Mount a Fixed Disk File System, located on the Fixed Disk Management menu under Disk, to mount a fixed disk file system.



Caution – It is strongly recommended that you use `sysadm chkfdfs` and `sysadm chkhdfs` to perform a file system check before attempting to mount file systems.

Mounting a Diskette-Based File System

`sysadm` automatically mounts a diskette-based file system when it is first created. Any time you use the diskette after that, you will need to explicitly mount the file system that is on it. Follow these steps to mount a diskette-based file system:

1. From the Disk menu select Diskette Management. Then select Mount a Diskette File System from the File System Management menu. Alternatively, you may type `sysadm mntfdfs` from the command line. Your screen will look similar to this:

```
Disk      File      Machine  Software  User      Help      Quit
Make a file system on a diskette accessible to users
SYSTEM ADMINISTRATION
Current Disk Usage
Diskette Management  ->
Mount a Diskette File System
Diskette Types
disk0_1.2M
disk0_360k
File System Mount Options
Mount point:  /mnt
Access allowed: <read-only >
OK          CANCEL      HELP
```

2. In the first field, select the drive and type of diskette you want to use. The first option selects a high-density diskette in the first (5.25-inch) diskette drive; the second option selects a double-density diskette in the first (5.25-inch) diskette drive. The choices displayed on your screen will depend on what is available on your system.
3. In the next field, select the directory in which you want to mount the file system. The default directory is `/mnt`, but you can specify a different directory by typing the name in this field.
4. In the third field, specify the type of access you want to allow on the file system by toggling the spacebar and selecting `read/write` or `read-only`.
5. Select the `OK` button at the bottom of the form. (If you want to cancel the operation, select the `CANCEL` button.)
6. When the system is mounted, you will be asked whether you want to mount another file system. If you do not want to mount another file system, select the `NO` button. If you do, select the `YES` button and another form will appear.

Now that the file system is mounted, you may access it on the diskette using an ordinary INTERACTIVE UNIX System path name.



Caution – You should always *unmount* the file system before removing the diskette from the drive. Refer to “Unmounting a File System” later in this chapter for information on how to do this.

Mounting a File System on the Second Fixed Disk

Although fixed disk file systems are automatically mounted when the system is booted, at times you may need to mount one explicitly, for example, if you unmounted a file system to check or repair it. (Refer to “Unmounting a File System” and “Checking and Repairing a File System” in this chapter for information about unmounting and checking file systems on fixed disks.) A file system on a fixed disk can be mounted using the `sysadm` command. Before you can mount a file system, you must know two things:

- The name of the device that contains the file system
- The directory name of the file system “mount point”

Refer to Chapter 22, “Understanding INTERACTIVE UNIX System File Systems,” for device file naming conventions. You will use the same device file name the system used when it created the file system at the time of system installation or when adding a second fixed disk, as described in the *INTERACTIVE UNIX System Installation Guide*.

1. To mount a new fixed disk file system, access the Fixed Disk Management menu of the Disk menu and select Mount a Fixed Disk File System. Your screen will look similar to this:

```
Disk      File      Machine  Software  User      Help      Quit
Make a fixed disk accessible to the system and its users
SYSTEM ADMINISTRATION
Current Disk Usage
Diskette Management  ->
Fixed Disk Man       Mount a Fixed Disk File System
Qu
Remap Disk
Add a Fixe
Check a Fi
Display Fi
Mount a Fi
Unmount a
Quit
Current File System Parameters
Device name:
Mount point:
Access allowed:  <read/write>
OK      CANCEL  HELP
```

2. The first field asks for the name of the device the file system resides on. Type the name of the device that contains the file system, for example, /dev/dsk/c0t0s4.
3. The next field asks for the point where you want to mount the file system. Type the name of the mount point in this field.
4. The third field allows you to specify read/write or read-only access for the file system. Select the appropriate option, then press Enter.
5. Select the OK button when you have completed the form. You will be asked whether you want to mount another file system. If you do not want to mount another system, select the NO button. If you do, select the YES button and another form will appear.

Unmounting a File System

Before you can remove a diskette from the drive, you must *unmount* or deactivate it. You must also unmount a fixed disk partition that you do not want to be a part of the active file system.

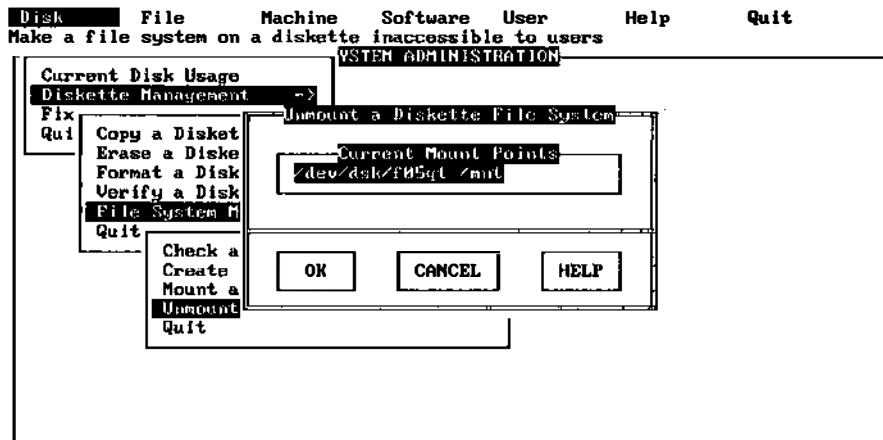
Unmounting a Diskette-Based File System

You can unmount a diskette file system using the `sysadm umntfdfs` command. If there are system users or programs using any directory under the mount point of this file system, the unmount will fail.

1. Change to any directory not on the diskette, for example, the root directory:

```
# cd /
```

2. Access the File System Management option of the Diskette Management menu under Disk, then select Unmount a Diskette File System. Your screen will look similar to this:



- This form shows the diskette file systems that are mounted. Select the file system you want to unmount, then press the OK button. The system will display another form that gives you the option of unmounting another diskette file system, if you wish. Select YES to unmount another diskette file system, and another form will appear. Select NO if you do not want to unmount another file system.

Unmounting a File System on a Fixed Disk

You can unmount a file system on a fixed disk using the `sysadm umnthdfs` command. If any system users or programs are using any directory under the mount point of this file system, the unmount will fail.

- Access the Fixed Disk Management menu of the Disk menu and select Unmount a Fixed Disk File System. Your screen will look similar to this:

```

Disk      File      Machine  Software  User      Help      Quit
Make a fixed disk inaccessible to the system and its users
SYSTEM ADMINISTRATION
Current Disk Usage
Diskette Management  ->
Fi
Qu
Remap Disk D
Add a Fixed
Check a Fixe
Display Part
Mount a Fixe
Unmount a Fi
Quit

          Unmount a Fixed Disk File System

          Current Mount Points
          /dev/dsk/c0t0s5  /home2
          /dev/dsk/c0t0s4  /home

          OK          CANCEL          HELP

```

- This form shows the file systems that are currently mounted. Select the file system you want to unmount, then press the OK button. The system will display another form that gives you the option of unmounting another fixed disk file system, if you wish. If you do not want to unmount another file system, select the NO button at the bottom of the form. If you do, select YES and another form will appear.

Checking the Space Available on a File System

You can find the disk space available for each file system on your computer with the `df` command. To use the `df` command, type the following at the system prompt:

```
# df
```

The file system mount point, special device, available space, and available inodes (which determine the number of new files that can be created) will be displayed for each mounted file system. You can also use the `df` command with a file system argument and display the used and available disk space for that file system. Refer to *df(1M)* for additional information.

Checking and Repairing a File System

The INTERACTIVE UNIX Operating System has several reliability features built in. Every time a file is modified, the system performs a series of file system updates. These updates, when written to the disk, produce a consistent file system.

Every time the INTERACTIVE UNIX Operating System is booted, your computer checks the status of each file system. If the system was not shut down cleanly, the `fsck` program is run. 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. If the information it checks does not match, it detects the inconsistency and attempts to repair it. Because `fsck` runs automatically on any file system with inconsistencies when the system is booted, you should typically not have to run `fsck` manually.

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. For more information about `fsck`, refer to *fsck(1M)* and to the *INTERACTIVE UNIX System Maintenance Guide*.

File System Corruption and Increasing Reliability

A file system can become corrupt in a variety of ways. Three of the most common ways are:

- Improper system shutdown and startup
- Removing media before unmounting its file system
- Hardware failure

To help keep your file systems reliable:

- Never remove a diskette while the diskette drive is on (the access light on the diskette drive is lit).
- Never remove a mounted diskette; always unmount it *before* you remove the diskette.
- Always use the shutdown command to bring the system down. This will unmount all file systems and ensure file system consistency.

Checking a File System

Use the File System Management `chkdfs` option under Diskette Management under the Disk menu to run `fsck` on damaged diskettes. Use the `chkdfs` option on the Fixed Disk Management menu under Disk to run `fsck` on a damaged file system on a fixed disk.

If an inconsistency in the file system is found, `fsck` displays a message describing the problem. 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 it encounters without assistance from you. In some cases the system asks you to confirm the action it plans to take. Let `fsck` repair as much of the file system as possible. Type `y` when `fsck` asks if it should proceed with a repair, or press Enter to accept the defaults. If `fsck` cannot repair the problem, you may have to reinstall a backup version of the file system.

You may find that badly damaged file systems that cannot be repaired can sometimes be mounted read-only, by using the `-r` flag from the command line or filling in the appropriate `sysadm` form with the read-only field set.

Checking a Diskette File System

To check the integrity of a diskette file system, use `sysadm`.

1. Access the Disk menu, then select the Diskette Management option. Access the File System Management option, then select Check a Diskette File System. The system displays a form similar to this:

```

Disk  File  Machine  Software  User  Help  Quit
Check a diskette file system for problems or errors
SYSTEM ADMINISTRATION
Current Disk Usage
Diskette Management ->
Fix
Copy a Di
Erase a D
Format a
Verify a
File Syst
Quit
Che
Cre
Mou
Unm
Qui
Diskette Types
disk0_1.44k
disk0_720k
disk1_1.2M
disk1_360k
Check mode:
<report errors only>
OK  CANCEL  HELP
    
```

2. In the first field, select the option that corresponds to the type of diskette in the drive you want to use. The first two options select a high- or low-density 3.5-inch diskette in the first diskette drive; the third and fourth options select a high- or low-density 5.25-inch diskette in the second diskette drive. (Note that other options may appear on your screen depending on what type and how many diskette drives you have installed on your system.)
3. In the Check mode: field, you can instruct the system to repair any errors that it detects or, alternatively, to simply report any errors.
4. Select the OK button when you have completed the form. The system will display another form that gives you the option of checking another diskette file system. If you do not want to check another file system, select the NO button. If you do, select the YES button and another form will appear.

Checking a Fixed Disk File System

Follow these steps to check the integrity of a fixed disk file system:

1. Use `sysadm` `harddisk` to access the Fixed Disk Management menu under the Disk menu.



Caution – Before checking the integrity of a fixed disk file system, it is important to make sure all other users are off the system. The checking process starts immediately after you select the appropriate option, so you must warn users *before* you start this `sysadm` process.

2. Select Check a Fixed Disk File System. The system displays a screen similar to this:

```

Disk      File      Machine  Software  User      Help      Quit
Run the fsck program to check and, optionally, repair the fixed disk
SYSTEM ADMINISTRATION
Current Disk Usage
Diskette Management ->
Fixed Disk Mana    Check a Fixed Disk File System
Qu
  Remap Disk
  Add a Fixed
  Check a Fix
  Display Fix
  Mount a Fix
  Unmount a F
  Quit
  Current File System Parameters
  Device Name:
  FS Type:
  Check mode: <report errors only>
  OK  CANCEL  HELP
  
```

3. In the first field, type the device name of the file system to be checked, for example, `/dev/dsk/c0t0s4`.
4. The next field is an information field filled in by the system. You cannot modify this field.

5. In the next field, instruct the system to repair any errors it detects or, alternatively, to simply report any errors.
6. Select the **OK** button. The system will display a form that gives you the option of checking another file system, if you wish. If you do not want to check another file system, select the **NO** button. If you do, select the **YES** button and another form will appear.

Once you have started to use your system, you will need to *back up* the files on your fixed disk (or disks) on a regular basis. The copying of files to removable media (such as a diskette or cartridge tape) is called a *backup*. This minimizes the possibility of lost data caused by hardware failures or operator error. It is important to perform backups on a regular basis because these copies are your only protection in the event of a file system crash.

There are two kinds of backups. An *archival 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 onto a diskette, tape, or other medium for storage.

After performing a backup, keep the removable media in a secure place so that there are always up-to-date duplicates of the files on your fixed disk. If the system should become corrupted or a file is mistakenly removed from the fixed disk, the corrupted or missing files can be copied from the removable media to the fixed disk. The procedure of copying the backup files from the removable media to the fixed disk is called a *restore*.

Use the `sysadm` File menu to transfer data from your fixed disk to another storage medium, usually a diskette. This chapter explains how to back up file systems, individual files, or directories onto diskettes.

Before You Back Up Your System

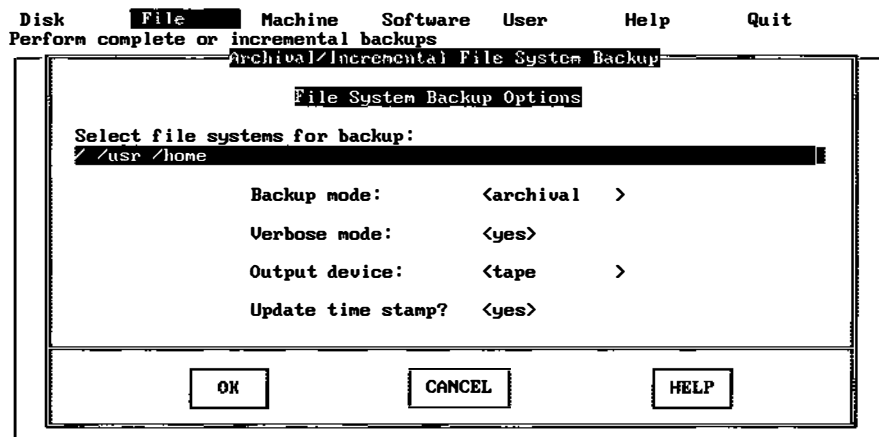
When you back up a file system, you must have available either a supply of formatted diskettes, or if your system is capable of backing up onto tape, a supply of cartridge tapes. (Note that you do not need to format tapes for backup.) Depending upon the size of your file systems, a single backup can require many diskettes or tapes. Be sure you have enough formatted diskettes or tapes available to complete the backup procedure before you run the backup procedure.

To format a diskette, use the Format a Diskette option of the Diskette Management menu under Disk. You will be guided through the procedure step-by-step.

Backing Up a File System

Follow these steps to back up a file system:

1. Access the sysadm File menu. Select Back Up or Restore Files, and then select the option Archival/Incremental Backup. Your screen will look similar to this:



2. This form contains the following fields:

Select file systems for backup:

The first field in this form shows the file systems available for backup. You can edit this field to show only the file system(s) you want to back up at this time.

Backup mode:

This field allows you to specify whether you want to perform an archival backup or an incremental backup. An archival backup is a complete backup of the file system. An incremental backup is a partial backup of all files that have changed since the date of the last timestamp. You should perform an archival backup the first time you use the backup option. Remember, when you do an archival backup of a file system, all the files on the specified file system are transferred to diskettes or tape. This may take some time to complete.

Verbose mode:

This field allows you to specify whether you want verbose mode enabled or not. If you select *yes*, the system will list files as they are backed up. If you select *no*, the system will not list the files.

Output device:

This field asks you to specify the output device to be used. Press the spacebar to pop up the list of output devices available on your system, and select the appropriate one. If the device you want is not listed, select the other option. A new form will appear that prompts you for the name of the output device on which you want to back up your files. Note that if you have a high-density diskette drive, you can back up onto either low- or high-density diskettes.

Update time stamp?

This field allows you to specify how far back incremental backups will go. Each time you perform a backup, you can either update the timestamp to reflect the time of the current backup or leave the timestamp unchanged. When an incremental backup is performed, the system compares the current timestamp with each file's date of modification and backs up all files that have been modified since that time.

3. When you have selected the backup options, select the OK button at the bottom of the form.
4. Next the system asks you to insert your first backup volume (a diskette or tape) in the drive and press Enter. The system copies all of the files on the named file system onto the backup volume. If more than one volume is required, you are prompted to insert additional volumes.
5. When the backup procedure is finished, label the backup volumes with the date of the backup and the name of the file system (and disk, if you are backing up multiple fixed disks).

The first file the system creates is an archive file, which lists all of the files to be backed up. This provides a convenient way to identify the files included in the backup.

Backing Up Individual Files and Directories

There may be occasions when you do not want to transfer a complete file system to a diskette. The Back Up Selected Files option may be used to transfer a file, a group of files, or the contents of a directory to a diskette.

1. Access the sysadm File menu and select the Back Up or Restore Files option. Select the option Back Up Selected Files. 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 these files:  [EDIT LIST]      Exclude these files:  [EDIT LIST]

Additional (Optional) Selection Criteria

Select files larger than:      bytes
Older than:      days      Newer than:      days
Owner:      <      >      Group:      <      >

[OK]      [CANCEL]      [HELP]
```

The form includes a number of fields that allow you to tailor your backup to include only the files and directories that you specify.

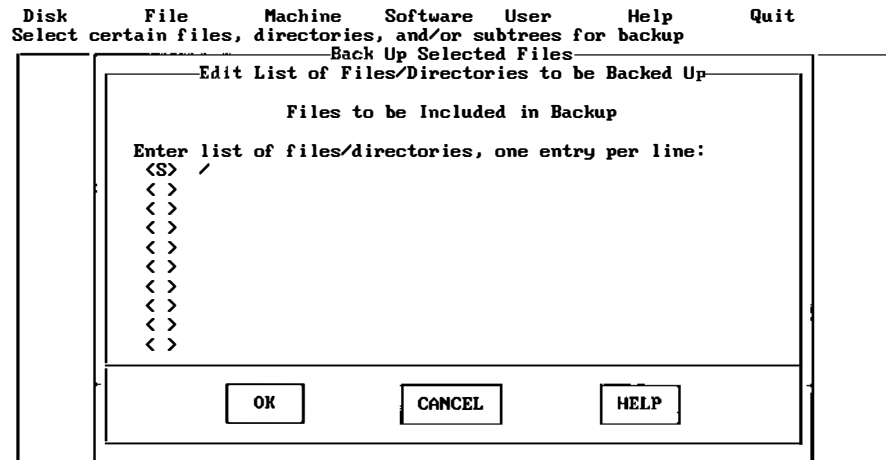
Starting directory:

This field allows you to specify a starting directory for the backup procedure. If you specify a starting directory, you can then use relative path names in the EDIT LIST fields described below.

Include these files:

This field allows you to specify the names of files, directories, and subtrees that you want included in your backup. If you have

specified a starting directory, you can use either relative or full path names in this field. Access the EDIT LIST in this field by pressing Enter. Your screen will look similar to this:



The first part of each line in this display allows you to specify whether the entry is a file, a directory, or a subtree. Use the spacebar to toggle through the choices (F, D, S, -). A *subtree* is a hierarchical arrangement of files and directories. If you select F, it indicates a single file. If you select D, only the files within the named directory will be included. If you select S, both files and directories within the named directory will be included in the backup procedure. The - symbol allows you to delete an entry. To delete an entry, select the - symbol and the field will be cleared.

Make your selection, then type in the name of the file or directory you want to include. Enter as many file or directory names as you want, one per line, then select the OK button at the bottom of the form to return to the previous form (Select Files for Backup).

Exclude these files:

This field allows you to exclude particular files, directories, or subtrees from the backup procedure. Access the EDIT LIST in this field and follow the same procedure you used to include files. Select

the appropriate code (F, D, or S), and enter the file or directory names, one per line. Select the OK button when you have completed the list.

Select files larger than: bytes
Older than:
Newer than:
Owner:
Group:

These fields in the backup display allow you to specify additional selection criteria for files. You can instruct the system to back up files that are of a certain size or age, or files that belong to a particular owner or group. For example, you might specify a range of files between 30 and 60 days old by specifying 30 in the Older than field and 60 in the Newer than field. When you have entered all pertinent specifications, 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 certain files, directories, and/or subtrees for backup
-----Back Up Selected-----
                               Select Backup Options
Output buffer size:  <      >      Verbose mode:      <yes>
Use ascii headers:  <yes>      Verify backup:      <no >
-----Specify Destination - Device/File/(Remote) Command-----
Archive destination type:  <device >
Output to:  <tape      >

[OK]          [CANCEL]          [HELP]
```

2. This form provides a number of options for backing up specific files that you select. You can specify the output buffer size, whether you want the backup procedure to report during the process, whether you want ASCII headers on your files, whether you want the system to check that the

archive media is readable, what type of backup is to be performed, and what the output device is to be. Toggle the spacebar to pop up the choices in each field and select the appropriate one for your backup. For help with a particular field, press F1 in the field. When you have completed the form, select the OK button.

3. The system first creates an archive file, which lists all of the files to be backed up. When the backup process is finished, label the backup medium with the date of the backup and the name of the directories and files.

Restoring Files

You will need to restore files in the event of hardware or user errors. Your users should be aware that some data may be lost, because any modifications made to the files since the time of the last backup will not be reflected in the restored files.

1. Access the sysadm File menu. Select the Back Up or Restore Files option, then select Restore Files. 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: <tape >
Destination directory: /
OK          CANCEL     HELP
    
```

This form includes these fields:

Archive source type:

In this field specify the archive source. Pop up the list of choices in this field and select the appropriate type of archive source. Select device if you are restoring from a tape or a diskette, file if you are restoring from an archive file, and command if, for example, you are restoring across a network.

Input from:

In this field specify the input source you are restoring from. Pop up the list of choices and select the appropriate input source. If you are restoring from diskettes, select the diskette drive and type of diskette you are using. If you are restoring from an archive file or using a command, select other and a new form will appear.

Destination directory:

This field allows you to specify where you want your files restored. If you do not specify a path name for the restore procedure, the root directory (/) is used.

2. If you are restoring from a device, make sure the diskette or tape containing the first volume of the backup is in the drive, then select the OK button at the bottom of the form. 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 these files:	EDIT LIST		Exclude these files:	EDIT LIST		
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		

3. This form allows you to specify which files you want to have restored. The first five fields are similar to those used in the backup procedure. Enter the names of files you want to include, the names of files you want to exclude, specify an output buffer size, enable or disable verbose mode, and indicate whether you want ASCII headers. (Refer to “Backing Up Individual Files and Directories” earlier in this chapter if you need more information on these fields.)

Create directories if necessary?

In this field, specify whether you want the system to create directories if necessary. Select `yes` if you want the system to create directories that are not on the fixed disk, if necessary for the restore procedure.

Keep the original date/time?

In this field, specify whether you want the files restored with the same date and timestamps they had when they were backed up. If you specify `no`, the files will have the current date and timestamp.

Restore only if newer?

In this field, select `yes` to specify that you want to restore only files that have a more recent date and timestamp than the equivalent files on the fixed disk. If you select `no`, the system will restore all specified files, regardless of the date and timestamp.

4. Select the `OK` button at the bottom of the form. The system will now proceed with the restore operation.
5. When the files are copied, the system will display another form that gives you the option of restoring more files, if you wish. If you do not want to restore more files, select the `NO` button. If you do, select the `YES` button.

What Are Devices and Device Drivers?

In the context of your computer system, a *device* is generally defined as a piece of hardware. For example, a diskette drive, a printer, or an on-board modem are all considered devices. For each device that you use with your computer system, you must have an associated software program called a *device driver*. The device driver is responsible for communicating between the hardware and the application program or operating system. It ensures that the device receives and carries out the instructions given by the user. Drivers for your devices can be supplied with the operating system, with the application program, or by the manufacturer of the device.

The INTERACTIVE UNIX Operating System is delivered with device drivers that support a wide variety of devices, such as commonly used terminals and printers. These drivers support your hardware in its standard configuration and allow you to easily add new devices. Some of these device drivers, such as the line printer driver, are configured into the kernel by default. If you plan to add a device that is already *supported* in this way by the INTERACTIVE UNIX System, it is not necessary to install a new device driver. See the *INTERACTIVE UNIX System Installation Guide* for details about which drivers are already configured into the kernel. To add a terminal or printer to your system, refer to the *INTERACTIVE UNIX System Maintenance Guide*.

Several device drivers that are available with your operating system are not configured into the kernel but are in the Additional Drivers subset.

The Kernel

The *kernel* is the core of the INTERACTIVE UNIX Operating System and all UNIX-based operating systems. It is the program that is responsible for managing the hardware and software resources of the computer. It manages user logins, runs programs, and keeps track of input to and output from devices. The kernel is stored in an ordinary file (`/unix`) and is tailored to a specific hardware and software configuration. You loaded the standard INTERACTIVE UNIX System kernel when you installed your INTERACTIVE UNIX Operating System.

If you plan to add new pieces of hardware to your system, in most cases you will have to reconfigure your system to support the new devices. If the type of device you plan to add is not already supported in your standard hardware configuration, then you must also reconfigure the system kernel. This process builds a new kernel that includes the driver for the new device.

SunSoft, Inc. has made this process as easy for you as possible by providing two system enhancements. First, a friendly, menu-driven interface called `kconfig` is provided to help you configure your kernel. `kconfig` is used to configure, build, and install new kernels. (For additional information about `kconfig`, refer to the *INTERACTIVE UNIX System Maintenance Guide* and to the `kconfig(1)` manual entry.) Like `sysadm`, `kconfig` provides step-by-step procedures that are designed to make it easy to perform a common set of system administration tasks. Second, the *High Performance Device Driver* is provided to increase the range of supported devices. It is discussed in the following section and in the *INTERACTIVE UNIX System Maintenance Guide*.

The High Performance Device Driver

The INTERACTIVE UNIX System kernel contains a set of integrated software packages called the High Performance Device Driver (HPDD). The HPDD allows several controllers to share a single driver, making it easier for you to add and use different types of controllers. It also significantly improves the speed of your system during most operations.

The HPDD supports most AT-compatible fixed disk controllers, many popular SCSI host adapters (including cartridge tape and CD-ROM (compact disc read-only memory) support), and a RAM disk. (A RAM disk is created by reserving a portion of the computer's available memory, which is then treated as if it were a disk storage device. Refer to the *INTERACTIVE UNIX System*

Maintenance Guide for more information about RAM disks and for a list of controllers supported by the HPDD.) Each time you add, remove, or change any of *these* devices, you must change your system configuration by:

- Configuring the HPDD to support the new configuration
- Rebuilding and installing the kernel
- Physically installing the hardware (except in the case of the RAM disk)

Note – There are two exceptions to this. If you are replacing a standard AT® controller of one type with a different standard AT controller, you do not need to reconfigure the HPDD. (Note that the hardware configuration of the two controllers must be the same.) Also, if you are simply adding a fixed disk drive to a controller that you have already configured, you may not need to reconfigure the HPDD.

You will use the `kconfig` program to reconfigure the HPDD.

These procedures are discussed in the *INTERACTIVE UNIX System Maintenance Guide*.

Tailoring 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
- One serial communications port (COM1)
- Up to three parallel ports
- A real time clock
- CMOS RAM

Basic INTERACTIVE UNIX System kernel, including:

- MS-DOS file system service
- UNIX System V file system service (S51K and S5L file system types)

- Common interprocess communication routines, including:
 - Interprocess communication message facility
 - Interprocess communication semaphore facility
 - Shared memory
- 287 or 387 floating point coprocessor emulator

Your vendor may deliver a different default configuration. Check the documentation supplied with your system to determine its default configuration.

You will need to reconfigure and rebuild your system kernel to:

- Add a new device and its driver
- Reconfigure the High Performance Device Driver
- Remove a device and its driver
- Add or remove certain software packages
- Change the system memory size
- Add or change tunable parameters

You will use the `kconfig` utility to configure, build, and install a new kernel after you make any of the changes listed above. Refer to the *INTERACTIVE UNIX System Maintenance Guide* to learn how to reconfigure and rebuild your system kernel.

Security deserves special consideration on a multi-user system. The information you have stored on your computer belongs to you and no one else. It is a valuable resource that requires protection. Three security features provided for your computer are:

- System backups
- Access permissions
- Passwords

Sources of Potential Damage

To provide adequate security for your system, you first need to consider the kinds of problems that might arise. It is important to consider every possible contingency: theft of the hardware, breakdowns in the hardware or software, tampering with data in the computer by unauthorized people, theft of data, and simple human error. To determine the types of problems to which your system may be vulnerable, ask yourself the following questions:

- *Are you using your computer in a large or a small office? Is it possible for someone to steal your computer?*
Prevent theft by setting up your computer in a secure place.
- *How many people have ready access to your system?*
Restrict the number of people with access to your computer by using passwords.

- *How sensitive or valuable is the information you are storing?*
Restrict the number of people who have access to sensitive data using the access permissions.
- *Do you transmit data over telephone lines?*
Limit access to your computer telephone numbers.

How you answer these questions will help you decide the measures you want to take to safeguard your data.

Basic Precautions

The following safeguards are recommended to everyone for ensuring the security of their computer and data:

- Set up your computer in an area that can be secured when you are not using it.
- Never leave your computer terminal logged in and unattended.
- If you are using your computer in an office environment, restrict the number of people who have access to the computer.

System Backups

If a breakdown occurs in the system, you may lose information that you had stored in the computer. You can avoid this type of loss if you have copies of your files stored on removable storage devices. Copy your data periodically onto diskettes or cartridge tapes and store them in a safe place, away from the computer site. Refer to Chapter 25, “Backing Up Files” for more information about the backup procedure.

Access Permissions

If you are sharing your computer with other users, you may want to share some information while keeping other information private. The INTERACTIVE UNIX Operating System allows you to satisfy both of these needs by letting you define:

- The users that have permission to access data

- The types of permission they have (that is, how they are allowed to use the data)

The INTERACTIVE UNIX System recognizes three categories of users of stored data: the owner of the data, the group to which the owner belongs, and all other people who use the system. Users can be given permission to use data in any or all of three ways: to read, write, and/or execute it. Refer to “Changing Permissions (chmod)” in Chapter 9, “Other Frequently Used Commands,” for more information about access permissions.

Whenever you create a file or directory, the system automatically identifies the users who will be allowed to read, write, and/or execute the file and the users who will be allowed to access the directory. As the owner of the file, only you and the superuser (usually the system administrator) can change the access permission after it has been created.

One way to restrict access to your files is to change the permission modes. Refer to Chapter 9, “Other Frequently Used Commands,” *chmod(1)*, and *ls(1)* for more information about changing access permissions.

To keep people from looking at the contents of a directory, you can deny execute permission to that directory. Others will be prevented from listing or looking at the contents of your directory.

Whenever a file or directory is created, the permission modes are automatically set for everyone to have access, unless the *umask* command is set in your system profile or personal *.profile*. You can use the *chmod* command to change the permissions each time you create a file, or you can use the *umask* command to change the default permission modes of a file that you will be creating. Individual users can place the *umask* command in their *.profile*, or you can alter the system default profile, */etc/profile*. Type:

```
$ umask nnn
```

nnn represents three octal digits that refer to read, write, and execute permission for owner, group, and other, respectively. The value of each permission digit is subtracted from the corresponding permissions digit. For example, if you add *umask 022* to */etc/profile*, all files normally created with *777* permission will now be created with *755* permissions. (*umask 022* is

set by default in `/etc/profile` in the INTERACTIVE UNIX Operating System.) A file created with 666 permissions will be created with 644 permissions. See `umask(1)` for additional information.

Password Administration

If you are at all worried about security, it is a good idea to force users to maintain passwords. When a login is first established, a password can be assigned, or one can be assigned later. The superuser can assign or change a password for any login. For more information about the password files and their management, refer to Chapter 19, “Managing User Accounts.”

Glossary



absolute path name

See **full path name**.

administrative login account

A login account used to give ordinary users restricted access to system management functions that must be performed frequently.

append

To add text to the end of an existing file.

ANSI

American National Standards Institute.

application

A tool, program, or window that provides the user with particular capabilities, such as sending electronic mail, printing files, or interacting with the operating system.

archival backup

A backup procedure that is used to back up all of the files and directories on a particular file system.

argument

A string of text that accompanies a command and gives the computer additional information to modify the result of the command.



ASCII file

An ordinary text file. ASCII stands for “American Standard Code for Information Interchange,” and refers to the way characters are represented in a computer.

AZERTY

The name used to reference French keyboard layouts.

back up

To copy files to removable media (such as a diskette or cartridge tape). A backup is the spare copy of data or software that you keep in case the original is damaged or lost. A backup may be incremental or archival. See **incremental backup** and **archival backup**.

background

Not in the foreground, that is, not interfering with current work on the terminal. A program that is executing in the background allows the user to continue to issue commands and interact with the computer system. To execute a program in the background, type an ampersand (&) at the end of the command.

batch mode

A dossette interface that permits commands to be invoked noninteractively in shell scripts.

baud rate

The speed at which a device that is connected to a tty line operates.

block files

Files that reside in `/dev/dsk`, which reference block devices, such as fixed disks, that access hardware one block (sector or record) at a time.

boot

To load an operating system or a standalone program into memory and execute it. This term comes from the saying, “to pull yourself up by your own bootstraps.”

Bourne shell

The Bourne shell is considered the standard UNIX System shell and is available on UNIX System V.3. It was written at Bell Laboratories and released by AT&T (now USL). See **shell**.

C shell

A UNIX System shell written and released by the University of California at Berkeley. It has gained widespread popularity, particularly with programmers. See **shell**.

case sensitive

Distinguishes between uppercase and lowercase.

character files

Files that reside in `/dev/rdsk`, which reference character devices, such as terminals, which generally process data one character at a time.

codepage

A **codeset**. This term is used in the DOS world, particularly by IBM.

codeset

A convention describing one-to-one relationships between symbols and numbers. It represents letters as numbers that can be stored in a computer's memory.

command

An instruction the user gives to the computer. The command is interpreted by the user interface, which instructs the computer to run the program that will perform the task requested by the command. See also **argument** and **option**.

command interpreter

A command interpreter passes the commands you enter to the operating system for processing and delivers the results to you. See also **shell**.

command line

The complete instruction the user gives to the computer, including the command name, options, arguments, and pipes.

comment

A part of a file that is not processed by the program that reads the file.

compose sequence

A special key or sequence of keys used to put the keyboard into a special mode where the system expects two more characters to be typed by the user before a character is generated. The default Compose key sequence for the INTERACTIVE UNIX System is Control-Shift-F1.



configuration files

Files that store information that affects the environment of an individual user or affects the system on a global basis. Many INTERACTIVE UNIX System configuration files are shell command scripts or contain shell variable assignments.

console terminal

The console terminal is comprised of the screen and keyboard directly attached to the Central Processing Unit (CPU). It is important for the system administrator to use the console terminal since most error messages generated by the system are displayed on the console terminal screen.

current working directory

The directory in which you are located. Its path name is returned when you use the `pwd` command.

cursor

A cursor is a pointer on your screen to where the next action will take place. When typing, the cursor indicates the position where the next character will appear. When using a TEN/PLUS function, the cursor indicates the line or paragraph on which that function will operate.

cursor-positioning functions

The functions that permit you to move the cursor from one position to another. Examples are `TAB`, `-TAB`, `↑`, `↓`, `←`, `→`, and `ENTER`.

cursor-positioning keys

The arrow keys on your keyboard that are usually used to move the cursor when editing text.

daemon

A program that runs as a background process to handle UNIX System activities, for example, file transfers, command executions, and cleanup routines.

deadkey

A procedure for overprinting invented by typewriter manufacturers, where when one key is pressed, a character is printed but the typewriter carriage does not move until the second key is pressed, so that characters consisting of two separate characters, such as `ê`, can be formed. The INTERACTIVE UNIX System `ttymap` utility can be used to assign deadkeys. The only difference is that when the first key is pressed, nothing happens until the second key is pressed, after which the entire character appears on the screen.

decimal representation

Another method of generating characters. When the Compose key is followed by three digits, the character that is internally represented by the three-digit number (in decimal) is generated.

default

The alternative chosen by the system when no choice is specified by the user.

device

In the context of a computer system, a device is generally defined as a piece of hardware. For example, a diskette drive, a printer, or an on-board modem are all considered devices.

device driver

A software program associated with each specific device that is responsible for communicating between the hardware and the application program or operating system. It ensures that the device receives and carries out the instructions given by the user. Drivers for devices can be supplied with the operating system, with the application program, or by the manufacturer of the device.

device-specifier

An alphabetic character followed immediately by a colon, which is used under DOS and with `dossette` to indicate a particular diskette drive or fixed disk partition. The device specifier `b:` usually indicates the second diskette drive.

directory

A special type of file that contains other files and/or directories. A typical directory contains related documents, such as memoranda or monthly sales reports.

editor

A utility or program written to facilitate the modification and manipulation of text files.

environment variable

A word relating to the user's shell environment that is assigned a value and has a special meaning to the INTERACTIVE UNIX System. For example, the default terminal type is a variable that is usually assigned a value for each user.

**error message**

A message from the operating system that appears on your screen in response to a problem the computer has encountered with a command.

escape sequence

A sequence of characters, such as Escape (the code generated by the Escape key) `o p`.

execute permission

The third permission in each set of file or directory permissions. Execute permission allows a user to execute a file or search a directory.

field

A location in a file or form that is reserved for a single piece of information. For example, a personnel file might include separate fields for an employee's name, address, Social Security number, and so on. Each TEN/PLUS directory screen has two fields, `File` and `Description`.

file

A document or a collection of information stored on the computer. For example, a file could be a list of phone numbers, a memorandum, or a report.

file access permissions

See **permissions**.

file system

A complete directory structure that is contained on a diskette or a fixed disk.

filters

Programs that accept their input from one source, such as the standard input, perform their appointed task on the data, and then write their results to the standard output without changing the input file in any way.

flag

An option to a command. See **option**.

full path name

The full path name of a file completely describes the location of the file in the system. It is the sequence of directories from the `root` directory to the file or directory you wish to reference. It consists of a slash (`/`), followed by one or more directory and file names, separated by slashes.



full-screen editor

A text editor that shows the changes you are making to a file on the screen as you make them.

function

A way of performing an often complex operation with one or more keystrokes, which usually saves you from having to perform a series of steps.

group

A group is a collection of users. Permission for a group refers to the second set of three permissions for a file or directory. These apply to members of the owner's group only.

hardware

The physical components of a computer. Examples include the keyboard, the screen (sometimes called the display or the monitor), and the printer.

hexadecimal

Base 16 numbering using 0–9 and A–F. Also known as “hex.”

hierarchical file system

A directory structure that is arranged in a ranked series, with a single master directory at the top level and additional levels of directories or files defined beneath it.

High Performance Device Driver

A set of integrated software packages that are part of the INTERACTIVE UNIX System kernel. Also called the HPDD, for short. The HPDD allows several controllers to share a single driver, making it easier to add and use different types of controllers and significantly improves the speed of the system during most operations.

home directory

An area of the file system assigned to a user. Most systems automatically place you in your home directory when you log in.

host

The principal computer that links to a system of terminals or computers.

I18N

Internationalization. See **internationalization**.



incremental backup	A backup procedure used to back up only those files and directories that have changed since the last complete backup.
interactive	To carry on a dialogue with the computer.
internationalization	Making a computer, a computer system, or a computer program function appropriately in a non-U.S. environment. See I18N .
ISO	The international standards organization. (Note that ISO is not an acronym.)
I/O redirection	The capability of specifying the standard input and output at the time of executing a command.
kernel	That part of the UNIX Operating System that performs the system management aspects.
Korn shell	The Korn Shell combines the features of both the Bourne shell and the C shell . See shell .
L10N	Localization. See localization .
line editor	A text editor that requires that you explicitly specify the line or lines you want to edit and does not show the changes you make to your file as they occur. A line editor is useful if you are using a hard copy terminal, i.e, one where you do not have a screen.
localization	The adaptation of computer programs to a single language and/or country. See L10N .
log in	To type your login name and password onto your computer or terminal, indicating that you are ready to gain access to your information.



log out	To terminate your access to the system. On most systems, this is accomplished by holding down the Control key while simultaneously typing d.
login account	The information stored in a computer that provides authorization for a person to use that computer's resources. The computer's system administrator usually sets up login accounts.
manual entry	A technical description and summary of use for UNIX System commands, system calls, subroutines, file formats, miscellaneous facilities, and special files.
mount	A system administration procedure used to attach a file system to other directories of the main system. A file system must be mounted in order to be accessible to users.
mount point	The directory where a file system will be installed.
multi-tasking	Capable of running many different processes (programs) at the same time.
multi-user	Capable of supporting more than one user at a time. On a multi-user computer system, each user has his or her own terminal plugged into the computer, and all can share its data and resources at the same time.
octal	The numbering system of the base 8 that uses the numbers 0, 1, 2, 3, 4, 5, 6, and 7.
operating system	The operating system is part of the system software. It is a collection of instructions written in some machine or programming language that tells the computer how to manage its operations, process and execute the user's requests, and run application software.
option	An optional argument that is available to modify the results of a specific command. Options are usually preceded by a dash (-).

**ordinary login account**

Every user on the system uses an ordinary login account to perform most tasks. Ordinary logins include a login ID, a password, and an area of the file system assigned to the user, called a home directory.

other

The third set of three permissions for a file or directory, which applies to everyone on the system who is neither the owner nor in the owner's group.

output mapping

Modification of the code sent by the system or the application to the screen before a character is displayed.

overwrite

To perform an operation on a file that deletes its contents and replaces it with the output of the operation. For example, if you `cat` a file and use the name of an already existing file as its destination, the contents of the original file is replaced with the contents of the new file.

owner

The user who creates a file and has control over the file's permissions.

parent directory

The directory immediately above the one in which you are located.

partitions

The fixed disk is divided into sections called partitions. You may have only one partition that takes up the entire fixed disk, or you may have several partitions, each containing its own operating system, such as the UNIX System or DOS.

path name

The sequence of directory names and (optionally) a file name, separated by slashes, that describes the location of a file or directory on a UNIX System. There are two types of path names, the full path name and the relative path name. The full path name consists of the sequence of directories from the `root` directory to the file or directory you wish to reference. The relative path name omits the `root` directory and directory names up to the current directory.

pattern matching character

A wildcard character. A wildcard character is used to match a character or a string of characters in a file or directory name. It is another type of shortcut.



permissions

Permissions determine who can read, write, or execute a file or directory. They are indicated by the letters *r*, *w*, and *x* obtained when you list the contents of a directory using the long option.

pipeline

To take the output of one program and direct it to be the input to another program.

pipes

Connections between two or more UNIX System commands, indicated by a vertical bar symbol (`|`).

popup box

A small box on your screen that displays instructions or error messages.

program

A set of instructions for a specific set of tasks, written in a machine or programming language, that tells the computer how to perform the tasks.

prompt

A symbol that displays on the screen to indicate that the system is ready to receive your commands. You may execute a command or run an application when the prompt is displayed.

protection mode

The permissions on a file or directory, which determine how it is “protected” from the owner, group, and others. See also **permissions**.

queue

As a noun, a line or list formed by items in a system waiting for service. As a verb, to put in a line for processing, such as a print queue.

QWERTY

The name used to reference U.S. English keyboard layouts.

QWERTZ

The name used to reference German keyboard layouts.

RAM

Random Access Memory.

read permission

The first permission in each set of file or directory permissions. Read permission allows a user to view the contents of a file or directory.



reboot	The procedure of bringing up the system.
redirect	To change the destination of the output or the input of a command.
relative path name	The path name of a file or directory, omitting the <code>root</code> directory and directory names up to the current directory.
restore	The procedure of copying the backup files from the removable media to the fixed disk.
ROM	Read Only Memory.
root directory	The main (top level) directory in the <code>root</code> file system, which is automatically created and given the unique name of <code>/</code> (slash) when your system is installed and initialized. All other directories in the system connect directly or indirectly to the <code>root</code> directory.
root file system	The permanent file system established on the first subpartition of the INTERACTIVE UNIX System partition on the first fixed disk. It contains all of the important files and subdirectories that are required to run the INTERACTIVE UNIX System.
root user	See superuser .
scroll	Move text off of the display so that you can see the preceding or following text. When you reach the bottom of the screen while typing, <code>TEN/PLUS</code> automatically scrolls the text forward, so that you always have a place for the next line. When you are reading a document, you can scroll text forward (up) one screen at a time by using <code>+PAGE</code> , or one-third of a screen at a time by using <code>+LINE</code> . Similarly, you can use <code>-PAGE</code> and <code>-LINE</code> to scroll text backward (down).

shell

That part of the UNIX System that interacts with the user. It is also called a command interpreter or user interface. The two major ones in use are the Bourne shell and the C shell. The Korn shell is becoming increasingly popular. The shell prompts you for commands, “interprets” your commands for the computer, and sees that it carries out the task you have requested.

shell program

See **shell script**.

shell prompt

The computer symbol that appears on your screen to indicate that the shell is ready to receive your commands. You may execute a command or run an application as soon as you see it.

shell script

A simple collection of shell commands normally executed on the command line, or a program that uses programming logic, variable assignment, and argument processing to perform complex tasks. Shell scripts are usually created by the user to perform frequently needed complex tasks.

single-user

Capable of supporting only one user at a time. On a single-user computer operating system, no one else can simultaneously share the computer’s data or resources. Most small personal computers are single-user systems.

software

The instructions that make a computer perform its functions. Software is divided into two main groups: application and system. Application software performs specific tasks, such as word processing, spreadsheets, educational programs, and games. System software includes operating systems that tell the computer how to run application software.

special files

Files that do not contain data. For example, device files are called special files because they are references to the actual programs that run the peripheral devices attached to your computer.

spooler

The term “spool” is an acronym for **simultaneous peripheral operations on-line**. The line printer spooling system allows you to send a file to be printed while you continue with other work.



standard input

Information coming from the keyboard, unless otherwise specified by the user. The user can specify that information come from a file, a device, or a pipe as well.

standard output

The destination of a program's data, considered to be the terminal screen, unless otherwise specified by the user. The user can specify that information be written to a file, a device, or a pipe as well.

string

A word, file name, command, or other sequence of characters or letters.

structured file

A special type of file available in the TEN/PLUS environment. TEN/PLUS keeps information about structured files that allows you to look at previous versions of those files.

subtree

A hierarchical arrangement of files and directories.

superuser

The most privileged user on the system (also called the `root` user). There are no restrictions imposed upon `root`, which means that the person logged in to the system with the `root` ID can access, modify, and delete every file and process on the system.

swap area

An area of the disk used to store portions of programs or data that have been temporarily swapped out of main memory.

system administrator

The individual responsible for installing, administering, and maintaining your INTERACTIVE UNIX System.

system dump

A procedure used by experienced system administrators to obtain detailed information about the state of the system when it crashed. (A system crash refers to the unexpected halting of the system, usually due to hardware or software failure.)



system login account

Accounts used to perform system administration tasks that require privileged access to the restricted files and directories on the system. The two most important system logins are `root` and `bin`.

system software

System software is composed of programs and utilities that manage the computer's resources and run application software. See **operating system**.

terminal

A physical device that consists of a monitor (screen), video adapter, and keyboard. It connects to a computer, which does the actual computer processing.

trigraph

A three-letter sequence used in an ANSI C source file that is interpreted as a single symbol. This is essential to the C language.

unmount

A system administration procedure used to detach a mounted file system from the main system and make it inaccessible to users.

user interface

The command interpreter, or that part of the operating system with which the user interacts.

utility

A program. Related utilities can be combined into a package that represents a specific application available with your computer.

variable

This term usually refers to a shell environment variable. A variable is a word that is assigned a value and has a special meaning to the INTERACTIVE UNIX System. For example, the default terminal type is a variable that is usually set for each terminal device.

virtual terminal

A feature that allows you to log in several times simultaneously on a console terminal and switch back and forth between the login sessions. Processes continue to run in the background even while out of view.

wildcard

A wildcard character is used to match a character or a string of characters in a file or directory name. It is a type of shortcut.



word wrap

A feature that eliminates the need for you to use Enter at the end of each line. When a word you are typing extends past the right margin, the entire word is moved to the left margin of the next line.

write permission

The second permission in each set of file or directory permissions. Write permission allows a user to modify (edit) the contents of a file or create and delete files in a directory.

Index

Symbols

- # symbol
 - as prompt, 171, 211
 - preceding a comment, 209
- \$ symbol, as prompt, 211
- & symbol, 69
- * symbol
 - as wildcard character, 63, 78
 - wildcard, caution, 64
- +LINE function
 - in TEN/PLUS, 115
- +PAGE function
 - in TEN/PLUS, 115
- +SEARCH function
 - in TEN/PLUS, 133
- .profile file
 - setting variables in, 213
- / symbol, 43
 - in path name, 237
 - in UNIX System path name, 102
- /dev/dsk
 - storing block device files in, 241
- /dev/rdisk
 - storing character device files in, 241
- /etc/group file, 198
- /etc/hosts file, 83
- /etc/passwd file, 197
- /etc/profile file, 212
- /mnt directory, 239
- /tmp file system, 244
- /unix file, 282
- < symbol, 65
- > symbol
 - in sysadm menus, 200
- > symbol, 65
 - as secondary shell prompt, 211
- >> symbol, 67
- ? symbol
 - as wildcard character, 61
- [. . .] symbol
 - as wildcard character, 64
- \ symbol
 - in DOS path names, 102
- | symbol, 68

Numerics

- 7-bit terminals
 - displaying data on, 157
- 8-bit codesets
 - as defined by ISO, 155

A

- access permission
 - denying, 73
- accessing a file system, 237
- account, login, 18
- adding a group, 204
- adding a user, 200 to 204
- administrative login account, 177
- alias, 83
- alternate file, 134
- appending to a file, 67
- argument
 - to a command, 27, 30
- arrow keys
 - cursor positioning with, 111
- ASCII codeset
 - IBM-extended, 153
 - supersets of, 153
- ASCII file, 114
- assigning a password, 203
- automatic login, 86

B

- background processing, 69
- backing up a file, 274 to 278
- backing up a file system, 272
- BACKSPACE function
 - in TEN/PLUS, 112
- batch mode
 - using dossette, 108
- bdftosnf utility
 - converting fonts with, 159
- BEGIN-LINE function
 - in TEN/PLUS, 132
- bin login, 179
- block files, 241
- booting the operating system, 18
- BREAK function
 - in TEN/PLUS, 134

C

- CANCEL function
 - in TEN/PLUS, 116
- case sensitive, 27, 45
- cat command, 33, 36
- category
 - controlling environment with, 159
- cd command, 50
- CDPATH variable, 210
- changing a password, 207
- changing directories, 50
 - in TEN/PLUS, 111
- changing margins, 132
- changing permissions, 74
- changing tabs, 133
- character classification table, 161
- character files, 241
- characters
 - generating non-English, 144
- checkfsys login account, 180
- chmod command, 71
- chown command, 77
- cleanup procedures
 - using crontab, 219
- CMOS RAM
 - setting time in, 215
- codeset
 - 7-bit, 151
 - converting with iconv, 157
 - selecting and configuring, 156
- collation order
 - user-specified, 162
- command
 - names, 28
 - options, 30
- command interpreter
 - shell, 20, 59
- command line, 27
- command mode, ftp, 91
- command modes, for line editor, 57
- command, definition, 27

commands
 cat, 33, 36
 cd, 50
 chown, 77
 cp, 52
 cpio, 254
 crontab, 219
 date, 32
 df, 266
 dossette, 107
 env, 214
 exit, 23
 ftp, 91
 kill, 82
 lp, 80
 ls, 29, 51
 mkdir, 49
 mv, 76
 passwd, 21
 pg, 79
 ps, 81
 pwd, 48
 rcp, 87
 rlogin, 85
 rm, 54
 rmdir, 54
 rwho, 90
 sort, 68
 telnet, 95
 umask, 287
 who, 31
 comment lines, 209
 Compose mode, 145
 compose sequences
 for non-English characters, 145
 configuration file
 sendmail, 227
 configuration files, 209
 configuring the kernel
 with kconfig, 282
 consistency, file system, 266
 console
 booting from, 18
 console terminal
 enabling virtual terminals on, 99
 for system administration, 168
 controller number, 242
 controllers
 with HPDD, 282
 conventions
 for data storage, 142
 conversion utilities, 103
 copying a file, 52, 123
 copying diskettes, 253
 copying files to a diskette, 254
 copying files with cpio, 254
 copying files with tar, 255
 copying text in a file, 121
 copying to a remote host, 87
 corrupted file system, 267
 cp command, 52
 cpio command, 254
 creating a directory, 49
 creating a file, 112
 creating a file system on diskette, 257
 creating directories
 in TEN/PLUS, 116
 cron program, 219
 crontab
 system cleanup with, 220
 crontab command, 219
 crontab file
 example, 219
 CUI help facility, 185
 CUI System
 menus and forms, 187
 current directory, 48
 cursor, moving, 132
 cursor-positioning functions, 111

D

- daemon
 - system account, 179
- data storage
 - conventions, 142
- data transfer
 - between types of machine, 142
- date
 - in international environment, 160
 - setting system, 215
- date command, 32
- deadkey
 - generating characters with, 145
- decimal representation
 - generating characters with, 146
- default
 - login group, 202
 - UID, 201
- default password, 22
- deleting directories
 - in TEN/PLUS, 122
- Description field
 - in TEN/PLUS, 111
- description section
 - in manual entry, 35
- destination host, 87
- device, 281
- device drivers, 241
- device files
 - file name examples, 245
 - naming conventions, 241
- device specifier, 102
 - using with `dossette`, 105
- `df` command, 266
- directory
 - changing, 50
 - changing owner of, 77
 - creating, 49, 116
 - deleting, 122
 - home, 21, 110, 196, 202
 - listing, 29, 52
 - naming conventions, 45
 - parent, 47
 - removing, 54
 - root, 43
- directory search path, 211
- Disk menu, 189
- disk partitions, 243
- disk space
 - checking with `df` command, 266
- diskette
 - copying, 253
 - copying files to, 254
 - device file name examples, 247
 - formatting, 252
 - naming conventions, 246
- diskette file system
 - mounting, 239
- diskette number, 246
- diskette-based file system
 - checking integrity of, 268
 - mounting, 260
 - unmounting, 264
- display server
 - with INTERACTIVE X11, 150
- displaying text, 79
- domain name, 229
- DOS file, 105
- DOS file system
 - unmounting, 102
- DOS files
 - converting to UNIX System format, 103
- DOS-FSS, 101
- `dossette` utility, 101
 - available commands, 106
 - batch mode, 108
 - interactive mode, 105
- drive name
 - in DOS path, 102
- driver, device, 281
- `dtou` conversion utility, 104

E

- ed line editor, 57
- editing files and directories, using
TEN/PLUS, 117
- editor
 - types of, 57, 58
- END-LINE function
 - in TEN/PLUS, 132
- env command, 214
- environment variables
 - defining, 210
- error messages
 - explanations of, 39
 - on console screen, 168
- escape sequences
 - related to function keys, 143
- European keyboard layout, 147
- execute permission, 71
- exit command, 23
- export statement
 - in `.profile` file, 214

F

- `fdisk` partitions, 242
- fields
 - as defined in forms, 188
- file, 110
 - alternate, 134
 - appending to, 67
 - backing up, 271, 274 to 278
 - changing owner of, 77
 - checking spelling in, 78
 - concatenating, 36
 - configuration, 209
 - conversion utilities, 103
 - converting to DOS format, 103
 - copying, 52, 123
 - copying text in, 121
 - creating, 112
 - deleting text in, 120
 - description of, 21
 - displaying contents of, 33

- moving, 123
- moving between DOS and UNIX
System, 101
- moving text in, 120
- naming conventions, 45
- overwriting, 53, 66, 76
- printing, 80, 134
- removing, 54
- renaming, 76
- restoring, 278 to 280
- sorting, 67
- viewing, 79

- file access permissions, 71
- file backup
 - as system security, 286
 - types of, 271
- File field, in TEN/PLUS, 111
- File menu, 190
- file name conventions
 - in TEN/PLUS, 111
- file system
 - backup, 272 to 274
 - checking inconsistencies in, 266
 - corruption of, 267
 - creating on a diskette, 257
 - fixed disk, 236
 - hierarchical, 43
 - mounting, 236, 239
 - naming conventions, 238
 - removable, 257
 - repairing with `fsck`, 266
 - storing user files on, 244
- file system, DOS
 - as part of UNIX System partition, 101
 - multiple users on, 101
 - unmounting, 102
- file transfer program, 91
- files, DOS
 - accessing from UNIX System, 101
- filter, 67
- fixed disk
 - device file names, 245
 - updating with `sync` program, 179

fixed disk file system, 236
 checking integrity of, 269
 mounting, 263
 mounting automatically, 260
 unmounting, 265
fixed disk partitions
 naming conventions, 242
flag, 30
font files
 for INTERACTIVE X11, 159
format conversions, 101
FORMAT function, 124
formatting a diskette, 252
ftp command, 91
ftp session, example, 93
ftp subcommands, 92
full path name, 45
full-screen editor, 57
function keys
 generating escape sequences
 with, 143

G

group ID, 196, 202, 204
group name, 204
 in /etc/group, 198
group, adding to the system, 204

H

HELP function
 in TEN/PLUS, 116
Help Index, 185
hierarchical file system, 43
high density diskette
 formatting, 252
High Performance Device Driver
 (HPDD), 282
home directory, 196, 202
 as part of login account, 177
 in TEN/PLUS, 110

HOME variable, 211
host name, 83
HPDD, reconfiguring, 283

I

I/O redirection, 65
I18N, as acronym for
 internationalisation, 142
IBM-codepages
 for non-English languages, 154
IBM-extended ASCII, 153
iconv utility
 converting codeset with, 157
ID
 group, 196, 202, 204
 login, 195, 196
 user, 196
IFS variable, 211
inodes
 changing default, 259
input
 redirection, 65
 standard, 65
input mode, telnet, 94
INSERT function
 in TEN/PLUS, 124
insert mode, 133
INSERT-MODE function
 in TEN/PLUS, 133
interactive mode, using dossette, 105
internal diagnostics
 running during rebooting, 175
international environment, 159 to 164
internationalisation features, 141 to 164
internet address, 83
interrupting a search, 134
ISO codesets
 partial list, 155

K

kconfig
 configuring the kernel with, 282
kernel, 59, 282 to 283
keyboard
 European and U.S., 143
 sections, 143
 special keys, 144
keyboard layout
 European and U.S., 147
 QWERTY, 144
keyboards
 AZERTY, 147
 on 7-bit terminals, 149
 QWERTZ, 147
 using toggle key with, 149
kill command, 82

L

L10N, as acronym for localisation, 142
lef conversion utility, 104
LEFT function
 in TEN/PLUS, 132
line editor, 57
 command modes, 57
-LINE function
 in TEN/PLUS, 115
links
 in help facility, 186
listing files and directories, 29
loadfont utility, 159
local environment, defined, 159
local network, 90
locale
 in international environment, 159
localisation of software, 142
LOCAL-MENU function
 in TEN/PLUS, 130
logging in
 to a virtual terminal, 99
 to TEN/PLUS, 109

logging out, 23
 of virtual terminals, 100
logical unit number, 243
login account, 18, 195
 types of, 177
login group, 202
login ID, 195, 196, 201
login program name, 197
login prompt, 196
login, automatic, 86
LOGNAME variable, 211
low density diskette
 formatting, 252
lp command, 80
ls command, 29, 51

M

Machine menu, 190
mail system
 configuration examples, 224
 configuring with sysadm, 226
MAIL variable, 211
MAILCHECK variable, 211
mail-related files
 location of, 225
makefsys, 180
manual entries
 format of, 35
margins, changing, 132
menu
 bypassing, 192
 Disk, 189
 File, 190
 Machine, 190
 Software, 190
 User, 190
MENU function, 128
menus, 116, 128
message of the day
 modifying, 218
mkdir command, 49

- modifier keys, on U.S. keyboards, 144
- monetary formatting
 - in international environment, 164
- mount point
 - creating with `mkdir`, 239
- `mount fsys` login account, 180
- mounting a diskette-based file system, 260
- mounting a file system, 236, 239
- mounting a fixed disk file system, 260, 263
- moving a file, 123
- moving screen to right and left, 132
- `mv` command, 76

N

- name line, in manual entry, 35
- name server, 230
- naming files, caution, 61
- network command format, 83
- network node name, 226
- network, local, 90
- networking commands, 83
- New Task Menu, 128
- news file
 - changing, 218
- node name conflict, 226
- numeric formatting
 - in international environment, 164
- numeric keypad, 143
- Numlock key, defined, 143
- `nuucp` system login, 180

O

- open subcommand, 91
- operating system
 - booting, 18
 - crash, 175
 - rebooting, 175
- option
 - to a command, 27, 30

- output
 - redirection, 65
 - standard, 65
- output mapping
 - with `ttymap`, 157
- overriding system defaults
 - with `.profile`, 213
- overwrite mode, 133
- overwriting a file, 53, 76
 - caution, 66

P

- PAGE function
 - in TEN/PLUS, 115
- parent directory, 47
- partition number, 243
- partitions
 - dividing disk into, 242
 - required and optional, 245
- `passwd` command, 21, 205
- password
 - aging, 205
 - assigning, 203
 - changing, 207
 - default, 22
 - guidelines, 205
 - setting, 21
 - used for login, 19
- passwords
 - as system security, 288
 - assigned to system logins, 178
- path name, 237
 - DOS syntax, 102
 - full, 45
 - relative, 45
 - shortcut (`.` and `..`), 46
- `PATH` variable, 211
- pattern matching character, 60
- permissions
 - changing, 71, 74
 - definitions, 74
 - octal equivalents, 73
 - on DOS file system, 105

- types of, 71
- user categories, 72
- pg command, 79
- PICK-COPY function
 - in TEN/PLUS, 121
- PICK-UP function
 - in TEN/PLUS, 118
- PID number, 69
- pipe, 68
- pipeline, using with lpr, 80
- popup boxes
 - in TEN/PLUS, 116
- port, 94
- port number, 94
- powerdown login, 169
 - using shutdown program, 24
- powerdown login account, 180
- PRINT function
 - in TEN/PLUS, 134
- printing a file, 80, 134
- process
 - checking, 81
 - stopping, 82
- process ID, 81
- process identification number, 69
- process status, 81
- processing, background, 69
- program, definition, 27
- programs, shell, 60
- prompt
 - shell, 20
 - types, 20
- protection modes, 71
- protocols, TCP/IP, 91
- ps command, 81
- PS1 variable, 211
- PS2 variable, 211
- pseudo-domain, 229
- public network
 - connecting to, 226

- PUT-DOWN function
 - in TEN/PLUS, 118
- pwd command, 48

Q

- QWERTY keyboard layout, 144

R

- radio buttons, 188
- RAM disk, 282
- rcp command, 87
- read permission, 71
- rebooting, 175
 - after shutdown, 18
- recreating files, in TEN/PLUS, 114
- redirecting I/O, 65
- relative path name, 45
- remote copying, 87
- remote host
 - executing a shell command, 88
 - listing users, 90
 - logging in to, 85
- remote login, telnet, 94
- remote machine, 83
- removable file system, 257
- removing a directory, 54
- removing a file, 54
- repairing a file system, 266
- resetting system date/time, 217
- restoring a file, 278 to 280
- rhost name, 83
- RIGHT function
 - in TEN/PLUS, 132
- rlogin command, 85
- rlogin session, terminating, 86
- rm command, 54
 - caution, 64
- rmdir command, 54
- root directory, 43, 235
- root file system, 235 to 236

root user, 171, 178
rwho command, 90

S

scripts, shell, 60
scrolling
 stopping and resuming, 33
scrolling a file
 in TEN/PLUS, 115
search
 interrupting, 134
 text, 133
-SEARCH function
 in TEN/PLUS, 134
sectors
 changing default, 259
 number per track, 246
sendmail, 223
 configuration file, 227
 tailoring parameters of, 230
SET-TAB function
 in TEN/PLUS, 133
setting a password, 21
setup program
 setting date and time with, 215
shell, 59, 197
 Bourne, 59
 C, 59
 command interpreter, 20
 features, 60
 Korn, 59
shell command
 executing on a remote host, 88
shell programs, 60
shell prompt, 20
 defining, 211
shell scripts, 60
shell variable, 209
shutdown
 rebooting after, 18
shutdown program, 24, 169
smart host, 232

smiling faces
 generating, 146
Software menu, 190
sort command, 68
sorting
 in international environment, 162
source host, 87
special files, 241
spell utility, 78
stopping a process, 82
string, 210
structured file
 in TEN/PLUS, 114
su command, 181
subdirectory, 110
superuser login ID, 171
swap area, 243
symbols
 in file names, 45
sync system account, 179
syntax
 of DOS path names, 102
sysadm
 CUI help facility, 185
sysadm hypertext links, 186
sysadm login account, 180
sysadm menus
 conventions for using, 191
sysadm program, 183
sysdemo account
 as tutorial, 184
system
 shutting down, 169
system administrator
 definition of, 167
 responsibilities of, 168
system cleanup
 with crontab, 220
system date/time
 resetting, 217
 setting, 215
system dump, after a crash, 175

system login, 177
 · assigning password, 178
system security
 access permissions, 286
 passwords, 288
 potential problems, 285
system time, 215

T

-TAB function
 in TEN/PLUS, 112
TAB function
 in TEN/PLUS, 112
tabs, changing, 133
tar command, 255
target number, 243
TCP/IP
 protocols, 91
telnet subcommands, 95
telnet utility, 94
TEN/PLUS
 +LINE function, 115
 +PAGE function, 115
 +SEARCH function, 133
 as user interface, 60
 BACKSPACE function, 112
 BEGIN-LINE function, 132
 BREAK function, 134
 CANCEL function, 116
 creating directories, 116
 deleting directories, 122
 Description field, 111
 editing functions, 117
 END-LINE function, 132
 File field, 111
 file name conventions, 111
 FORMAT function, 124
 HELP function, 116
 INSERT function, 124
 INSERT-MODE function, 133
 LEFT function, 132
 -LINE function, 115
 LOCAL-MENU function, 130
 logging in to, 109
 MENU function, 128
 New Task Menu, 128
 -PAGE function, 115
 PICK-COPY function, 121
 PICK-UP function, 118
 popup boxes, 116
 PRINT function, 134
 PUT-DOWN function, 118
 recreating file versions, 114
 RIGHT function, 132
 scrolling a file, 115
 -SEARCH function, 134
 SET-TAB function, 133
 structured file, 114
 -TAB function, 112
 TAB function, 112
 USE function, 134
 word wrap, 111
 ZOOM-IN function, 111
 ZOOM-OUT function, 111
TEN/PLUS editor, 58
TEN/PLUS functions, summary, 135 to
 139
TERM variable, 212
termcap interface, 143
terminal type
 defining, 212
terminfo interface, 143
text editing, 133
time
 displayed with date command, 32
 in international environment, 160
time zone
 defining, 212
time, system, 215
toggle key, in Russian-English
 keyboards, 149
transferring data
 between types of machine, 142
translation of software, 142
trigraphs, representing symbols with, 149

ttymap
 configuring a codeset with, 156
 output mapping with, 157
ttymap utility
 assigning deadkeys with, 145
tutorial login
 sysdemo, 184
TZ variable, 212

U

UID, 196
 default, 201
umask command
 changing permissions with, 287
umountfsys login account, 180
unmounting
 diskette-based file system, 264
 file system, caution, 262
 fixed disk file system, 265
USE function
 in TEN/PLUS, 134
user accounts
 accessing with su command, 181
user categories, 72
user ID, 19, 196, 201
User menu, 190
user name, 196, 201
users, listing of, 32
utod conversion utility, 104
uucp system login, 180

V

variables
 default environment, 210 to 212
 shell, 209
vi editor, 58
viewing a file, 79
virtual terminal
 switching, 99

W

warning message
 at system shutdown, 172
warning period
 at system shutdown, 173
who command, 31
wildcard
 expansion with DOS-FSS, 103
 using, 60
word wrap, in TEN/PLUS, 111
working directory, 47
working environment
 defining in /etc/profile, 212
write permission, 71

X

xttymap utility, 150

Z

ZOOM-IN function in TEN/PLUS, 111
ZOOM-OUT function in TEN/PLUS, 111

Part No.: 801-7871-10
Revision A of June 1994



A Sun Microsystems, Inc. Business

2550 Garcia Avenue
Mountain View, CA 94043 U.S.A.