Power Mach^{Ten}

User's Guide

TENON INTERSYSTEMS

New Dimensions in Personal Workstation Technology

1123 Chapala Street, Santa Barbara, CA 93101
TEL 805-963-6983 • FAX 805-962-8202
1-800-6-MACH-10

Revision 2
December 1995

TABLE OF CONTENTS

1.0 Power Mach^{Ten} — UNIX on the Macintosh Desktop 1

1.1 Power Mach^{Ten} 2
1.2 Mach^{Ten} PowerPC Architecture 2
1.2.1 Dynamic Memory Configuration 3
1.2.2 Dynamically Linked, Shared Libraries 3
1.2.3 Memory Mapped File Access 4
1.2.4 Integrated Software Development Tools 4

2.0 Installing Mach^{Ten} 5

2.1 Optional Installations 11
2.2 Performance Tuning — Optimizing Mach^{Ten} 11
2.2.1 System Optimization Guidelines 12
2.2.2 How Can You Tell When a System is Approaching Its Limits? 15
2.3 Reinstallation 15
2.3.1 Reconfiguration 16
2.3.1.1 Automated Reconfiguration 16
2.3.1.2 Manual Reconfiguration 17
2.3.2 Accessing Mach^{Ten} Files Directly From the CD-ROM 18
2.3.3 Accessing Online Man Pages Directly From the CD-ROM 18
2.3.4 Accessing Mach^{Ten} Sources Directly From the CD-ROM 19
2.3.5 Accessing Mach^{Ten} X Window Software Directly From the CD-ROM 19
2.4 Troubleshooting 19

3.0 The Mach^{Ten} Control Panel 21

3.1 Scheduling Priority Slide Bar 21
3.2 Configuration Screens 22
3.2.1 General Configuration Screen 22
3.2.1.1 Host Name 22
3.2.1.2 Time Zone 23
3.2.2 Memory Configuration Screen 23
3.2.2.1 Maximum Number Parameters 23

3.2.3 Networking Configuration Screen 24
3.2.3.1 Enable IP Forwarding 25
3.2.3.2 Enable Incoming Mail 25
3.2.3.3 Enable Incoming Connections 25
3.2.3.4 Internet (IP) Addresses 25
3.2.3.5 Netmasks 27
3.2.3.6 Maximum Transmission Unit Settings 27

4.0 Launching Mach^{Ten} 29
4.1 The Mach^{Ten} Login Console 32
4.2 Mach^{Ten} Windows 34
4.2.1 The Apple Menu 35
4.2.2 The File Menu 36
4.2.3 The Edit Menu 36
4.2.4 The Window Menu 37
4.2.4.1 The Positions Sub-Menu 38
4.2.4.2 The Size Sub-Menu 38
4.2.4.3 The Order Sub-Menu 39
4.2.5 The Fonts Menu 41

5.0 Mach^{Ten} Administration 43
5.1 A Word About Man Pages 43
5.2 Tailoring the Startup Environment 44
5.2.1 Setting Up Mach^{Ten} to Boot Automatically 45
5.2.2 Manual Startup 46
5.3 Tailoring Mach^{Ten} Windows 46
5.4 Tailoring Your UNIX Environment 46
5.5 Quitting Mach^{Ten} 47
5.6 Login Accounts 48
5.6.1 The Concept of a Home Directory and a User Environment 48
5.6.2 The UNIX Shell 48
5.6.3 Superuser and Privileges 49
5.6.4 A Word About Security 49
5.6.5 The Root Password 49
5.6.6 Setting Up User Accounts 50
5.6.7 The Password File 52
5.6.8 The Group File 53

5.6.9 Administrative Login Accounts 54
5.6.10 Writing to All Users 55
5.6.11 Disabling User Logins 55
5.6.12 Removing Users 55
5.6.13 Changing the Message-of-the-Day 55
5.6.14 Special Characters 56
5.7 Managing Your UNIX Environment 57
5.7.1 What's Running? 57
5.7.2 Killing a Program 57
5.7.3 Background Program Execution 57
5.7.4 Shell Files 58

6.0 The Mach^Ten File System 61

6.1 File System Structure 61
6.1.1 The root Directory Tree 61
6.1.2 Floating Root 62
6.1.3 The usr Directory Tree 62
6.1.4 The var Directory Tree 62
6.2 Major System Administration Files 63
6.3 File Names 64
6.3.1 Maximum Number of Characters 64
6.3.2 Case Sensitive File Names 64
6.3.3 Component Separators 64
6.3.4 Non-Printable Characters 65
6.4 Linked Files 65
6.4.1 Hard Links 65
6.5 Locked Files 66
6.6 Directory Link Counts 66
6.7 File Types 66
6.8 File Permissions 69
6.9 Time Stamps 69
6.10 Mounting Macintosh Volumes 70
6.10.1 Mounting Permissions 71
6.10.2 Automatic Mounting of Removable Media 71
6.10.3 Unmounting Macintosh Volumes 71
6.10.4 Automatic Unmounting 71
6.10.5 Formatting Floppies 72
6.11 Accessing Macintosh Files from UNIX Applications 72
6.11.1 AppleSingle Encapsulation 72
6.11.2 Differentiating UNIX and Macintosh Files 72

6.11.3 Utilities for Manipulating Macintosh Files 73
6.11.3.1 dfork and rfork 73
6.11.3.2 chtype and chcreat 73
6.12 Text File Manipulation 73
6.12.1 Combining Macintosh and UNIX Text 73
6.12.2 Text File Manipulation Utilities 73
6.12.2.1 mactext 73
6.12.2.2 unixtext 74
6.12.2.3 Unix <-> Text 74
6.12.2.4 dfork.text 74
6.12.3 Text Editing 74
6.12.4 BBEdit 75
6.12.5 UNIX to Macintosh "Copy and Paste" 75
6.13 File Maintenance 75
6.14 Archiving and Backing Up File Systems 76
6.14.1 Archiving to NFS 76
6.14.2 Tape Backups 77

7.0 The Mach^{Ten} Network File System (NFS) 79

7.1 Using NFS 79
7.2 How NFS Works 79
7.3 Setting Up an NFS Server or an NFS Client 80
7.3.1 Server Exporting 81
7.3.2 Client Mounting 81
7.3.3 Set-Up Summary 83
7.4 NFS Volumes on the Macintosh Desktop — Desktop NFS 84
7.4.1 Desktop NFS Volume Mounting 84
7.4.1.1 Automatic Mounting of Desktop NFS Volumes 85
7.4.2 Macintosh Record Locking on Desktop NFS Volumes 85
7.4.2.1 Bwnfsd UID and GID Mapping 86
7.4.3 Authentication Options 86
7.4.3.1 Credentials of Current User 86
7.4.3.2 Prompt for Username and Password 87
7.4.3.3 Username and Password on Command Line 87
7.4.3.4 User ID and Password on Command Line 87
7.4.4 File System Security 88
7.4.4.1 AppleShare Access Privileges 88
7.4.4.2 UNIX Access Privileges 88

7.4.4.3 Mapping AppleShare Access Privileges into UNIX Access Privileges 89
7.4.4.4 Access Privilege Strategies 90
7.4.4.5 Differences Between Apple File Sharing and DTNFS 90
7.4.5 AppleDouble 90
7.4.6 DOS Extensions 91

8.0 Printing 93

8.1 Mach^{Ten} Print Spooling 93
8.1.1 The Print Spooler Database 93
8.2 Local Printing 94
8.2.1 Printing Text Files to a PostScript LaserWriter on AppleTalk 94
8.2.2 Spooled Printing to an ASCII-Based ImageWriter Connected Directly to a Serial Port 94
8.3 Remote Printing 96
8.3.1 Spooled Printing to a Remote UNIX Printer 96
8.3.2 Receiving Remote Print Jobs 97
8.4 Selecting an Alternate Printer 97
8.5 Extending Remote Printing to Macintosh Applications 98
8.5.1 Lwsrv Configuration Files 99
8.5.1.1 Font Lists 100
8.5.1.2 PostScript Macros 100
8.5.2 Enabling lwsrv 100
8.6 Print Job Status Mail Notification 101
8.7 Status and lp Management Programs 102

9.0 Networking with Mach^{Ten} 103

9.1 Using Mach^{Ten} as an Internet Host 105
9.1.1 The Mach^{Ten} Networking Control Panel 106
9.1.2 Configuring Multiple Interfaces 106
9.2 Using Mach^{Ten} as a Domain Name Server 106
9.2.1 Configuring Mach^{Ten} as a Domain Name Server 107
9.2.2 The Domain Name Resolver 108
9.3 Using Mach^{Ten} as a Web Server 108
9.3.1 Importing Macintosh Image Files 109
9.3.2 Multihoming Your Mach^{Ten} Web Server 109

9.4 Using Mach^{Ten} as a Router 110
9.4.1 Invoking IP Forwarding 111
9.5 Using MacTCP Applications with Mach^{Ten} 111
9.6 Using Mach^{Ten} as a POP Mail Server 111
9.7 Using Electronic Mail 111
9.8 Using FTP 113
9.9 Using Mach^{Ten} as an Anonymous FTP Server 114
9.10 Using Telnet 114
9.11 Using a Distributed File System (NFS) 114
9.12 Using Berkeley r-series Commands 114
9.13 Using Serial Line Communications 115
9.13.1 Cabling and Connectors 115
9.13.2 Matching Serial Ports to UNIX Devices 115
9.13.3 Baud Rates Supported 116
9.13.4 General Purpose Interface (GPI) Serial Cable 117
9.13.5 Connecting a Modem to a Mach^{Ten} Serial Port 117
9.13.6 Connecting a Terminal 118
9.13.7 Configuring the System to Support an ASCII Terminal 118
9.13.8 Using PPP 118
9.13.8.1 Setting up Mach^{Ten} PPP on your Macintosh 118
9.13.8.2 Configuring Your Mach^{Ten} System as a PPP Server 119
9.13.8.3 Configuring Your Mach^{Ten} System as a PPP Client 121
9.13.8.4 IP Addressing Strategy for Multiple PPP Clients 122
9.13.9 Using SLIP 123
9.13.9.1 Configuring Mach^{Ten} To Dial-out To SLIP Servers 123
9.13.9.2 Tip Activity Log 123
9.13.9.3 The Tip Configuration Database 124
9.13.9.4 Hardware Flow Control 125
9.13.9.5 Tip Login Script 126
9.13.9.6 Header Compression 126
9.13.9.7 Setting a Default Route 127
9.13.9.8 Setting the MTU 127
9.13.10 Configuring Mach^{Ten} As A Dial-In SLIP Server 127
9.13.10.1 Serial Port Set Up 128
9.13.10.2 Login Authentication 128
9.13.10.3 The Sliplogin Program 129
9.13.10.4 SLIP Session Termination 129
9.13.10.5 Manual Connections To SLIP Servers 129
9.13.10.6 Logging In 130

9.13.11 Starting SLIP 130
9.13.11.1 Testing Connectivity 131
9.13.11.2 Stopping SLIP 131
9.13.11.3 C-Shell Scripts to Expedite SLIP Connections 132
9.13.12 Using Communications ToolBox Devices 133
9.13.12.1 Creating CommToolBox Devices 134
9.13.12.2 Configuring UNIX Communications Programs For CommToolBox Devices 134
9.13.12.3 CommToolBox Notes 136

10.0 Mach^{Ten} Programming Environment 137

10.1 Software Development Tools 137
10.2 Allocating Memory in Mach^{Ten} 139
10.2.1 Setting the Mach^{Ten} Heap Size 140
10.3 PEF and XCOFF 140
10.4 Shared Libraries 140
10.4.1 Shared Library Production 141
10.4.2 Run-Time and Compile-Time Libraries 141
10.4.3 Run-Time and Compile-Time Naming Conventions 141
10.5 Traditional UNIX Libraries 142
10.6 File System Hierarchy 142
10.7 Cross-Development Tools And Targets 143
10.7.1 Pre-Defined Names 144
10.7.2 Programming Notes 144
10.7.3 Default Mach^{Ten} Environment 144
10.7.4 Header Files 145
10.7.5 Makefiles 145
10.8 GCC 145
10.9 C++ 146
10.10 Fortran 146
10.11 Loader 146
10.11.1 mkpef 146
10.12 Symbol Information 147
10.13 Real Memory Issues 147
10.13.1 Calculating Memory Requirements 149
10.14 Making Macintosh Applications 149
10.14.1 Macintosh OS Interface Libraries 149

10.14.2 Macintosh OS Header Definition Files 150
10.14.3 Non-Standard Startup Routine 151
10.14.4 Macintosh Application Construction 151
10.15 Debugging 151
10.15.1 Debugging Using gdb 152
10.15.2 Non-Mach^{Ten} Debugging Tools 152
10.15.2.1 MacsBug 152
10.15.2.2 The Debugger 153
10.15.2.3 Macintosh Debugger for PowerPC 153
10.15.2.4 Metrowerks Debugger 153
10.16 Programming Example 154
10.16.1 Rogue 154
10.16.2 Building the Executable 154
10.16.3 Debugging Using MacsBug or Other Macintosh Debuggers 154
10.16.4 Finding the Program 154
10.16.5 Building a Symbol Table 155

11.0 The X Window System 157

11.1 The X Desktop 158
11.1.1 Starting the X Server 158
11.1.2 The Menu Bar 159
11.1.2.1 The File Menu 159
11.1.2.2 The Window Menu 160
11.1.3 Automatic Launch of the X Server 161
11.1.4 Quitting the X Server 162
11.1.5 Running the X Server Without a Menu Bar 162
11.1.5.1 Menu Bar Shortcuts 162
11.2 Administering the X Window Software Environment 163
11.2.1 Starting Clients 163
11.2.1.1 The Start Up Script 164
11.2.1.2 Resources: X Application Preferences 166
11.2.2 The Window Manager Client 170
11.2.2.1 Starting the Window Manager 170
11.2.2.2 Using the Mouse 171
11.2.2.3 Selecting a Window 171
11.2.2.4 Moving a Window 172
11.2.2.5 Changing the Size of a Window 173

11.2.2.6 Changing a Window Into an Icon 175
11.2.2.7 Moving an Icon 176
11.2.2.8 Restoring a Window From an Icon 176
11.2.2.9 Displaying a Window Menu and Making Selections 177
11.2.2.10 Summary of Window Menu Functions 178
11.2.2.11 Raising a Window 179
11.2.2.12 Quitting the Window Environment 180
11.2.3 The X Server Program 181
11.2.3.1 X Server Start Up Options 181
11.2.3.2 Mouse Button Mapping 184
11.2.3.3 Keyboard Mapping 186
11.2.3.4 Controlling X Client Access to the Server 187
11.2.3.5 Using an Alternate Server Display Number 188
11.2.3.6 Server Error Logging 188
11.2.3.7 X Server Performance Tuning Guide 188
11.2.4 Font Service 189
11.2.4.1 The Default Font Path 189
11.2.4.2 Obtaining Fonts From a Network Font Server 190
11.2.4.3 Providing Fonts Over the Network 190
11.2.5 X Display Management Under Mach^{Ten} 191
11.2.5.1 The Mach^{Ten} X Server and XDM 191
11.2.6 X Over Serial Lines 193
11.3 Mach^{Ten} X Window Software Overview 194
11.3.1 Preparing Your Macintosh Control Panels 196
11.3.2 Getting Started With X 199
11.3.3 Building X Applications 200
11.3.3.1 Running X Client Applications 201
11.3.3.2 The X11 Application Development Environment Under Mach^{Ten} 201
11.3.3.3 Programming Notes 203
11.4 X Window Software Frequently Asked Questions 203
11.5 References & Suggested Reading 207

LIST OF FIGURES

Figure 1:	Mach^{Ten} on the Macintosh Desktop	1
Figure 2:	Mach^{Ten} System Architecture	2
Figure 3:	Mach^{Ten} CD-ROM Contents	8
Figure 4:	Mach^{Ten} Folder	9
Figure 5:	Mach^{Ten} Console Window	10
Figure 6:	Control Panel Warning Dialog Box	20
Figure 7:	Mach^{Ten} Control Panel	21
Figure 8:	General Configuration Screen	22
Figure 9:	Memory Configuration Screen	23
Figure 10:	Networking Configuration Screen	24
Figure 11:	Example Network Configuration	28
Figure 12:	Mach^{Ten} Application Folder	29
Figure 13:	Mach^{Ten} Initialization	30
Figure 14:	Mach^{Ten} Login Console	33
Figure 15:	Mach^{Ten} Terminal Window	34
Figure 16:	Mach^{Ten} Windows	35
Figure 17:	user .cshrc File	51
Figure 18:	The /etc/password File	52
Figure 19:	The /etc/group File	53
Figure 20:	Special Keyboard Characters	56
Figure 21:	Using ps to Manage Your Environment	58
Figure 22:	Example of File Permissions	69
Figure 23:	Spooling Printer Output to UNIX Printers Using lwsrv	99
Figure 24:	Resource Sharing: FTP, mail, rlogin	105
Figure 25:	Example /etc/hosts	107
Figure 26:	General Purpose Interface (GPI) Serial Cable	117
Figure 27:	The X Desktop	158
Figure 28:	The File Menu	159
Figure 29:	The Window Menu	160
Figure 30:	The Order Sub-Menu	161
Figure 31:	The Motif Environment	170
Figure 32:	Motif Pointer Shapes	171
Figure 33:	Grabbing the Title Bar	172
Figure 34:	Repositioning the Window	172
Figure 35:	A Window and Its Elastic Outline	174
Figure 36:	The Title Bar Minimize Button	175
Figure 37:	The Motif Icon	175
Figure 38:	An Icon and Its Pop-up Label	176

Figure 39: The Restored Window 177
Figure 40: The Motif Window Menu 178
Figure 41: A Partially Concealed Window 179
Figure 42: The Window Revealed 180
Figure 43: Logitech MouseMan™ Control Panel 185
Figure 44: The Monitors Control Panel 196
Figure 45: The Mouse Control Panel 197
Figure 46: The Keyboard Control Panel 197
Figure 47: The Control Strip Control Panel 198
Figure 48: The Mach^{Ten} Control Panel 199

LIST OF TABLES

Table 1: System Optimization Guidelines 13
Table 2: Mach^{Ten} Daemons 31
Table 3: Mach^{Ten} Data Files 32
Table 4: Mach^{Ten} TCP/IP Protocol Stack (Comparison with MacTCP) 103
Table 5: Baud Rates Supported 116
Table 6: Places to Grab on the Window Frame 173
Table 7: Functions in the Window Menu 178

Preface

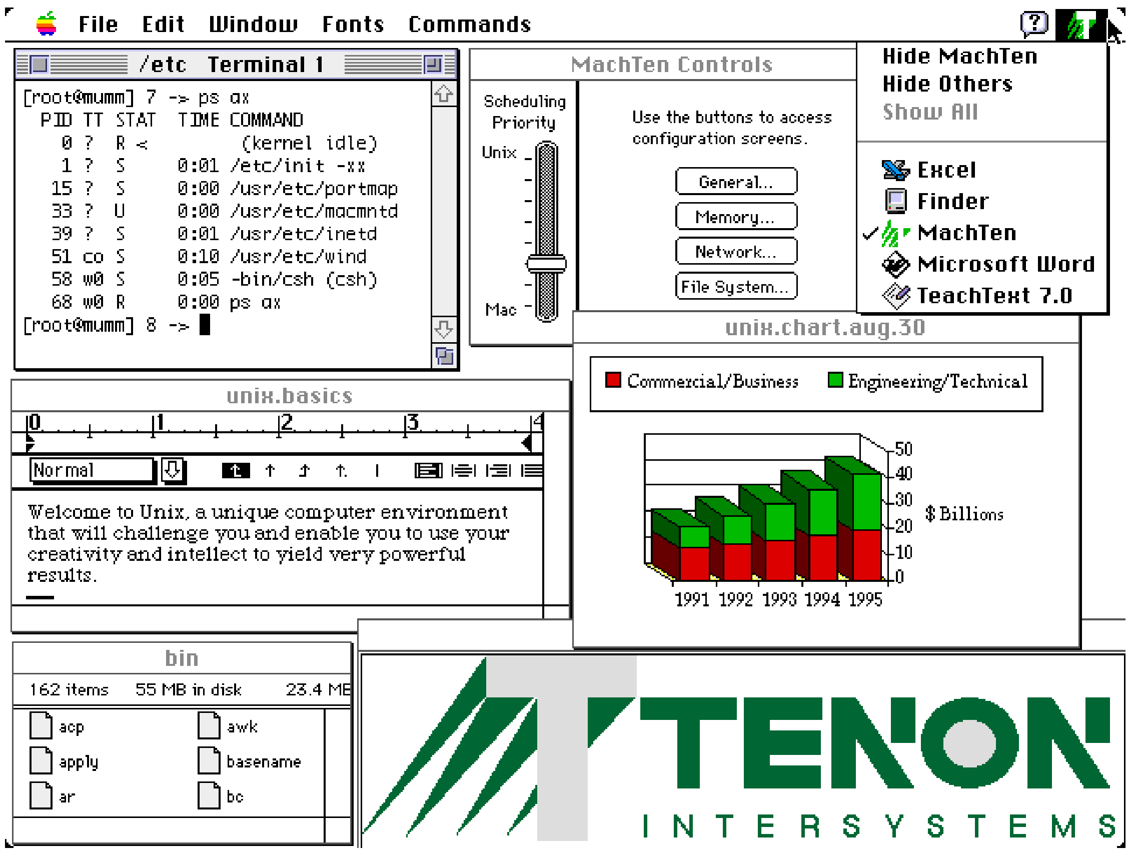
Mach^{Ten} is a complete UNIX system that runs as a Macintosh application. Mach^{Ten} provides a full suite of Internet services, distributed file system services, and remote printer spooling. These services are available not only to UNIX applications, but also to Macintosh applications. Mach^{Ten} also supports a sophisticated set of UNIX and X software development tools as well as a high-performance X Window server.

The combined strength of UNIX and X enables seamless integration of applications and resources across networks. Having UNIX and X on your Macintosh will enable you to benefit from the rich array of Macintosh easy-to-use applications and, at the same time, let you tap into your multi-vendor client/server environment to run remote applications and to share resources across your corporate network and the Internet.

This manual assumes some knowledge and understanding of UNIX. All levels of experience will easily be able to set up and use both Mach^{Ten} and the Mach^{Ten} X Window Software. If you are comfortable using UNIX, but are accustomed to having your local system administrator set up your UNIX environment and handle problems, the simplicity and accessibility of Mach^{Ten} may turn you into a UNIX wizard. If you are new to UNIX, you may want to supplement this User's Guide with UNIX tutorials. The supplemental UNIX papers included on the Mach^{Ten} CD provide more explanation and tutorial information on the care and feeding of UNIX. The Macintosh owner's guide that came with your computer should be consulted for Macintosh-specific information.

1.0 Power Mach^{Ten} — UNIX[†] on the Macintosh Desktop

The Macintosh "desktop" is created and maintained by a Macintosh system application called Finder™. When Mach^{Ten} is installed on your Macintosh, you still have the Finder desktop available, and you can easily switch from Macintosh applications to Mach^{Ten}. Mach^{Ten} and MacOS essentially operate as co-resident operating systems, sharing the processor and being enriched by each other's environment. Using Mach^{Ten} will enable you to work with Macintosh and UNIX programs concurrently, "copy and paste" between Macintosh and UNIX documents and applications, and share files, printers, and other peripherals with networked systems. Figure 1 shows a typical Mach^{Ten} desktop display.



```
[root@mumm] 7 -> ps ax
  PID TT STAT  TIME COMMAND
    0 ?  R <        (kernel idle)
    1 ?  S     0:01 /etc/init -xx
   15 ?  S     0:00 /usr/etc/portmap
   33 ?  U     0:00 /usr/etc/macmntd
   39 ?  S     0:01 /usr/etc/inetd
   51 co S     0:10 /usr/etc/wind
   58 w0 S     0:05 -bin/csh (csh)
   68 w0 R     0:00 ps ax
[root@mumm] 8 ->
```

Welcome to Unix, a unique computer environment that will challenge you and enable you to use your creativity and intellect to yield very powerful results.

Figure 1. Mach^{Ten} on the Macintosh Desktop

[†] UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

1.1 Power Mach^{Ten}

Mach^{Ten} is based on the University of California, Berkeley UNIX built on a Carnegie Mellon Mach kernel. The Mach kernel replaces many of the Berkeley UNIX internals and provides a number of new features not available in traditional UNIX systems. Mach^{Ten} both refines and extends the capabilities of the native Macintosh Operating System by providing a pre-emptive multitasking UNIX environment that coexists with the MacOS cooperative sharing execution environment. With Mach^{Ten}, Macintosh applications, UNIX applications, Mach applications and X applications run simultaneously. Mach^{Ten} represents the confluence of the best features of the Macintosh and the systematic power and elegance of UNIX and Mach.

1.2 Mach^{Ten} PowerPC Architecture

Figure 2 shows the basic system architecture. Power Mach^{Ten} consists of a family of shared libraries. The libraries are divided along functional lines that preserve or enhance the features of 68K-based Mach^{Ten}

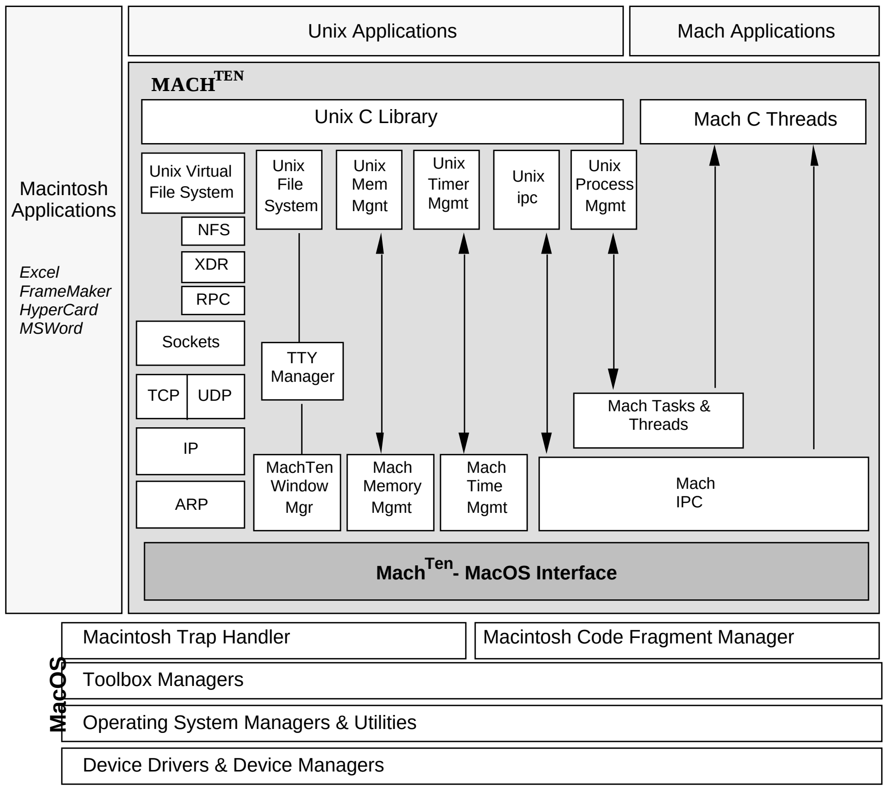


Figure 2. Mach^{Ten} System Architecture

systems. The kernel is a collection of shared libraries, the traditional UNIX libraries (*libc*, *libm*, etc.) are each a shared library, and every UNIX application is also a shared library. This organization takes advantage of the shared library, dynamic linking and memory mapped file access features and maximizes memory savings, as each software component is loaded into memory only once.

1.2.1 Dynamic Memory Configuration

Mach^{Ten} dynamically scales its memory requirements while the system is running. As more applications are initiated, the system brings in the necessary support so that the memory consumption grows as user requirements demand.

This memory configuration occurs along two axes: first, the system adjusts its memory data tables as each new application is initiated, with more applications requiring more memory; second, the system configures itself functionally, with each new application potentially requiring system components to be brought into memory. This continuous reconfiguration results in improved utilization of memory and processor resources and supports a range of application needs:

- small memory footprint — an internet domain name server or a network router
- intermediate memory footprint — mail or Web *httpd* server or X server with local X clients
- full memory footprint — high-end operations, full time sharing or software development, where multiple simultaneous operations are more important than maintaining a low memory profile.

1.2.2 Dynamically Linked, Shared Libraries

The PowerPC Executable Format (PEF) is based on the concept of dynamically loadable, shared libraries. In this architecture, software is composed of a private space and calls to shared libraries. When an application is created, a definition library is used to satisfy compiler header call definition requirements. When an application is executed, a runtime version of the library is used to satisfy library calls. If that library has already been loaded into memory, access is obtained to the runtime library through a dynamic linkage process. Otherwise, the library is loaded into memory and then dynamic linkage occurs. When the last client of a shared library has completed processing, the library is deallocated and its memory is reclaimed.

Compared to static linking used in traditional UNIX systems, dynamic linking results in much smaller binary images, reducing both system memory requirements and disk footprints.

1.2.3 Memory Mapped File Access

The Power Macintosh Memory Manager provides file mapped memory access for programs. Allocation of application instruction memory is deferred until an actual reference to a specific 'page' of instructions is made by the application. Only then is memory allocated and the specific 'page' of instructions copied into memory from disk. This is a significant benefit for large libraries that are only used a little-at-a-time.

1.2.4 Integrated Software Development Tools

Power Mach^{Ten} creates binary PEF files that integrate directly with other Macintosh development tools. Software produced by development systems from Apple, Metrowerks and others may be freely intermixed with the output of the Mach^{Ten} development system. With Mach^{Ten}, UNIX software development tools can be used to generate Macintosh applications and Macintosh software development tools can be used to generate UNIX applications. This enables developers to build hybrid applications that make both MacOS and Mach^{Ten} system requests. Such hybrid applications will be able to simultaneously take advantage of the power of the MacOS and UNIX.

2.0 Installing Mach^{Ten}

Mach^{Ten} will run on all Power Macintoshes and Power Macintosh clones, with MacOS version 7.1.2 or later. The entire Mach^{Ten} system is contained on a single CD-ROM in Macintosh HFS format and is comprised of these components:

- **Mach^{Ten} Root**

  Contains the Mach^{Ten} application, the Mach^{Ten} kernel and the UNIX libraries. The standard Mach^{Ten} application set includes network access programs, printer spooling, text editors, basic utilities, administration aids, and software development tools. The traditional UNIX software development tools — *gcc, ld, nm, ar,* etc. — have been ported to run in PowerPC native mode and produce Power PC native mode Code Fragments.

- **Mach^{Ten} System Folder Files**

  Files placed in the System Folder of your MacOS boot volume, including Mach^{Ten} Controls and Mach^{Ten} NFS.

- **Utilities**

  BBEdit 3.5. Mach^{Ten} includes BBEdit version 3.5, which is a complete distribution of the commercial version of BBEdit. (See the *Bare Bones Software* folder in the *Utilities* folder on the Mach^{Ten} CD for more information.) The default preferences for BBEdit 3.5 have been customized for Mach^{Ten}. These customized preferences are in the *Preferences* folder located in the *System Folder Items* folder on the Mach^{Ten} CD.

  Unix <-> Text. Mach^{Ten} includes the text file translation program *Unix <-> Text* for converting Mach^{Ten} text files from the UNIX text file format to the Macintosh text file format. (See section "6.12 Text File Manipulation" for more information.

- **Optional Components**

  X Windows. Power Mach^{Ten} includes a complete distribution of the X11R5 X Window software from MIT. This includes an X Server, an X client development environment, and some sample X clients.

  Man Pages. Mach^{Ten} manual pages are available in two formats. The traditional UNIX format is a *tty* or *cat* readable format suitable for use with the Mach^{Ten} *man* command. These *tty man* pages are Mach^{Ten} text files which may be read with Macintosh word processors and utilities. The PostScript man pages are suitable for printing to a laser printer.

Info Files. Some GNU applications do not provide traditional UNIX man pages. These applications use GNU "info" files to provide information on how these applications work.

Ported Applications. In addition to the standard Mach^{Ten} application set, several other significant applications have been ported to Mach^{Ten}.

Mach^{Ten} runs as an application, therefore all of the network and peripheral devices that your Macintosh supports are still supported when Mach^{Ten} is installed. Since Mach^{Ten} internet software has been interfaced to MacOS LocalTalk, MacOS EtherTalk, MacOS TokenTalk and MacOS serial port hardware, all of the Mach^{Ten} communications are equally applicable to LocalTalk, EtherTalk, TokenTalk or serial line networks, individually or in combination with each other.

Follow the steps below to install the complete Mach^{Ten} software package.

Step 1 — Site Preparation

- Verify RAM

  Pull down the Apple menu and choose "About the Finder" (or "About this Macintosh"). It will display the version of the System as well as the amount of RAM configured for the system. Make sure the "Total Memory" is at least 16MB[†], check MacOS version and "Power Macintosh".

- Verify disk space

  Make sure the hard disk has at least 45MB available by viewing any folder on your hard disk using "View by Icon". For the optional Mach^{Ten} X Window software you will need approximately 50MB of free space. If there is not ample free disk you will need to remove documents and folders to make additional space on your hard disk. Mach^{Ten} does not have to be installed on the MacOS boot volume.

NOTE: The free disk space requirement numbers are based on a target volume with an unformatted capacity of 160MB. Due to the MacOS block allocation scheme, large capacity disks will require more free space for the complete software distribution; smaller capacity disks will require less disk space.

A 500MB capacity disk will require up to 65MB for Mach^{Ten}. To conserve disk space, we suggest installing Mach^{Ten} on a file partition no larger than 250MB.

[†] 16MB RAM is recommended for good performance; Power Mach^{Ten} itself consumes about 4MB RAM.

- Prepare your hard disk

  The efficiency of the disk device driver plays an important role in the overall performance of the system. Reformatting your hard disk with a high performance device driver can noticeably improve overall performance.

  All MacOS system software, network drivers and other peripheral software should be installed before the Mach^{Ten} application is started.

  If you are installing Power Mach^{Ten} over an existing version of Mach^{Ten} on your disk, you may want to review section "2.3 Reinstallation".

- Assemble IP addressing and configuration information

  The installation process requires knowledge of your network environment. You will need the following information:

  - the local time zone
  - the name of your host (the machine on which Mach^{Ten} is being installed)
  - your Internet address on Ethernet (or EtherTalk) and the subnetwork mask (if appropriate)
  - your Internet address on AppleTalk and subnetwork mask (if appropriate)
  - your domain name and the Internet address of your Domain Name Server
  - the name and Internet address of your default gateway (you may be using either a MacIP gateway or a standard IP gateway)

  Mach^{Ten} includes its own TCP protocol stack. If you have already configured the Apple MacTCP Control Panel, Mach^{Ten} will use the networking information provided by MacTCP by default. Mach^{Ten} also works with Open Transport. In this case, if MacTCP is installed Mach^{Ten} still gets its configuration from MacTCP.

  If you are running Mach^{Ten} on a Macintosh that is not connected to a network, you can reconfigure Mach^{Ten} for the added network connections at a later time. Refer to section "2.3.1 Reconfiguration".

- Quit all applications

  Quit all currently running Macintosh applications. AutoDoubler and any startup Macintosh applications should also be disabled. Holding down the shift key while rebooting will disable virus protection software and startup routines that can interfere with the installation.

Step 2 — Installing the Mach^{Ten} Software on Your Hard Disk

- Insert the Mach^{Ten} CD-ROM and double-click on the disk icon. The disk will show an open window as illustrated below.

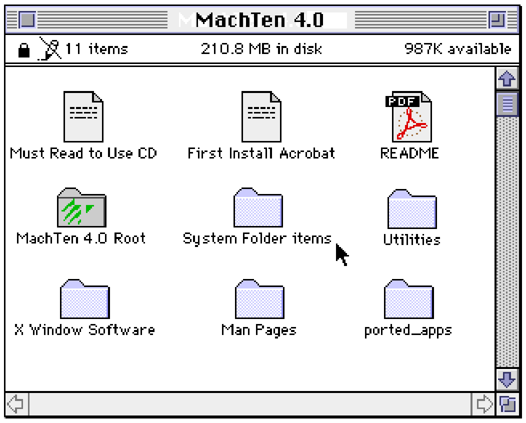


Figure 3. Mach^{Ten} CD-ROM Contents

The installation procedure requires using the Finder to drag and copy selected folders and files from the CD-ROM to your hard disk. The following steps detail the installation instructions.

- Drag the *MachTen 4.0Root* folder from the CD to your hard disk. This folder contains several thousand files which make up the Mach^{Ten} distribution. This will require approximately 45MB of disk space and may take several minutes.

- Make an alias to the *MachTen Libraries* folder located in the *MachTen* folder in the *MachTen 4.0Root* folder and put this alias in the *Extensions* folder in your *System Folder.* (Be sure to make this alias to the *MachTen Libraries* folder which is now on your hard disk, not the *MachTen Libraries* folder on the CD-ROM.)

- Drag *MachTen Controls* from the *Control Panels* folder (in the *System Folder items* folder) onto your *System Folder*. The Finder will automatically place this file in the appropriate location.

- Drag *MachTen NFS* from the *Extensions* folder (in the *System Folder Items* folder) to your *System Folder* only if you are going to use the Desktop NFS (DTNFS) under Mach^{Ten}. The Finder will automatically place this file in the appropriate location.

- Drag *BBEdit Prefs* from the *Preferences* folder (in the *System Folder items* folder) to the *Preferences* folder in your *System Folder.* These Preferences modify the commercial version of BBEdit to have file creator and default types that cater to Mach^{Ten}.

- Reboot

- Configure the Mach^{Ten} Control Panel according to the instructions in section “3.0 The Mach^{Ten} Control Panel”.

Step 3 — Launch Mach^{Ten}

- Launch Mach^{Ten} by clicking on the *MachTen* application in the *MachTen* folder in the *MachTen Root* folder. You may make an alias to this application and put it on the desktop for easier launching. If your system does not successfully start up and present you with the `login:` prompt after a moment, contact Tenon Technical Support.

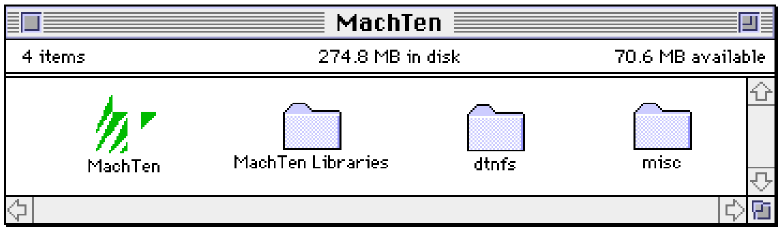


Figure 4. Mach^{Ten} Folder

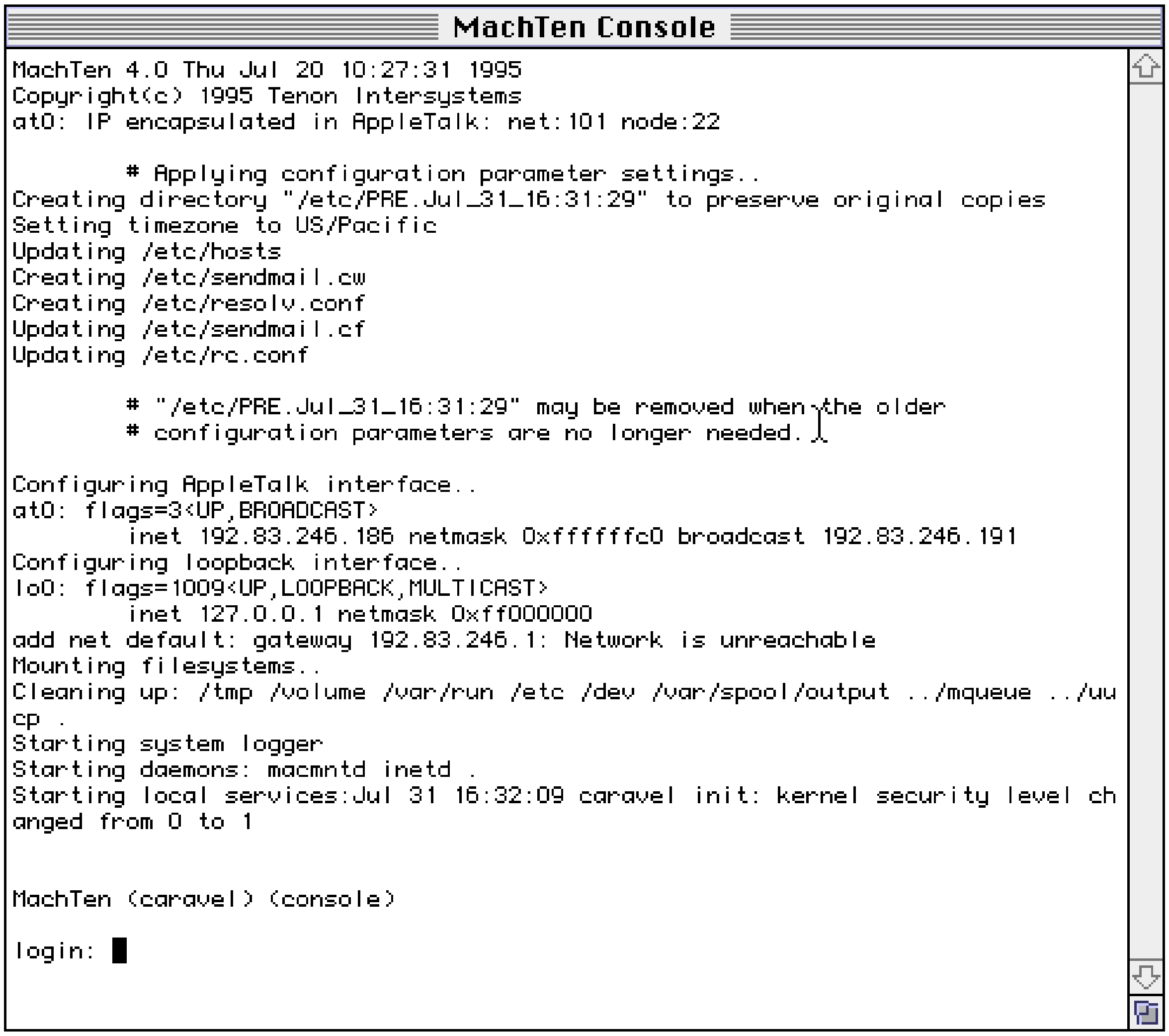
A Mach^{Ten} Console Window will appear with system configuration information taken from the Mach^{Ten} Control Panel.

Your Mach^{Ten} system is now ready to use. Please read section "4.1 The Mach^{Ten} Login Console" for information on how to log in. Refer to the following sections for additional set-up procedures: "3.0 The Mach^{Ten} Control Panel", "5.6.5 The Root Password" and "5.6.6 Setting Up User Accounts".

IMPORTANT: In the same manner that successfully booting and running the Macintosh OS is dependent on the *System Folder* and some very specifically named files and folders in the *System Folder*, Mach^{Ten} is dependent on its file system hierarchy, including the names, locations and permissions of its files.

Unless you are fully aware of Mach^{Ten}'s dependency on its file system organization and the read / write ability of the contained files, DO NOT reorganize the Mach^{Ten} distribution hierarchy.

Be very careful when using Macintosh imposed restrictions on the Mach^{Ten} folders and files. This includes locking or assigning File Sharing privileges to any Mach^{Ten} file or folder.



```
MachTen 4.0 Thu Jul 20 10:27:31 1995
Copyright(c) 1995 Tenon Intersystems
at0: IP encapsulated in AppleTalk: net:101 node:22

        # Applying configuration parameter settings..
Creating directory "/etc/PRE.Jul_31_16:31:29" to preserve original copies
Setting timezone to US/Pacific
Updating /etc/hosts
Creating /etc/sendmail.cw
Creating /etc/resolv.conf
Updating /etc/sendmail.cf
Updating /etc/rc.conf

        # "/etc/PRE.Jul_31_16:31:29" may be removed when the older
        # configuration parameters are no longer needed.

Configuring AppleTalk interface..
at0: flags=3<UP,BROADCAST>
        inet 192.83.246.186 netmask 0xffffffc0 broadcast 192.83.246.191
Configuring loopback interface..
lo0: flags=1009<UP,LOOPBACK,MULTICAST>
        inet 127.0.0.1 netmask 0xff000000
add net default: gateway 192.83.246.1: Network is unreachable
Mounting filesystems..
Cleaning up: /tmp /volume /var/run /etc /dev /var/spool/output ../mqueue ../uu
cp .
Starting system logger
Starting daemons: macmntd inetd .
Starting local services:Jul 31 16:32:09 caravel init: kernel security level ch
anged from 0 to 1


MachTen (caravel) (console)

login: █
```

Figure 5. Mach^{Ten} Console Window

2.1 Optional Installations

The following components are optional and may be installed at any time after Mach^{Ten} has been installed. Mach^{Ten} should ***not*** be running at the time these optional packages are installed.

- **X Window Software**

  Drag the *X11R5* folder from the *X Window Software* folder on the CD to the *usr* folder in your *MachTen Root* folder. To simplify the installation process, Mach^{Ten} places all of the X11 distribution in the */usr/X11R5* directory. The traditional paths to the X binaries and libraries (e.g., */usr/bin/X11* and */usr/lib/X11*) are still available in Mach^{Ten} via symbolic links already in place in the *MachTen 4.0 Root* hierarchy.

- **Man Pages**

  To install the *tty* readable *man* pages, drag all of the folders in the *tty* folder, which is in the *Man Pages* folder on the CD, to the */usr/share/man* folder in the *MachTen Root* on folder your hard disk. To install the postscript format man pages, drag all of the folders in the *Postscript* folder, which is in the *Man Pages* folder on the CD, to the */usr/share/man* folder in the *MachTen Root* folder on your hard disk. To install the GNU info files, drag the *info* folder in the *Man Pages* folder on the Mach^{Ten} CD to the */usr/share* folder in the *MachTen Root* folder on your hard disk.

- **Ported Applications**

  Pre-compiled binaries and a source distribution of popular public domain applications — such as *emacs*, *bash* and *perl5* — are contained in the *ported_apps* directory. Directions for making and installing these applications are contained in accompanying README files. Additional ported applications are online in Tenon's *ftp* server (*ftp.tenon.com*).

2.2 Performance Tuning — Optimizing Mach^{Ten}

Mach^{Ten} differs from other UNIX implementations in several significant ways. Mach^{Ten} was designed to run on the Macintosh platform. This includes machines with limited disk space and memory. As services and capabilities are added to these systems, it is inevitable that one or the other of these limits will be reached.

You can optimize Mach^{Ten} performance by tailoring your system to fit your particular hardware configuration and software service needs. Mach^{Ten} performance can be dramatically improved by giving some attention to the following configuration options.

- **Use a fast disk driver**

  The efficiency of the disk device driver plays an important role in the overall performance of the system. Choosing a fast disk and disk driver (such as FWB[†]) and adhering to the guidelines of disk optimizing tools will improve the Mach^{Ten} system's performance.

[†] FWB, Inc., 2040 Polk Street, Suite 215, San Francisco, CA 9109, 415-474-8055.

- **Use disk caching**

  Setting the disk cache in the MacOS Memory Control Panel is essential to getting the most performance from your Macintosh. MacOS file manipulation software operates much more efficiently when disk caching is enabled and set to a large value. In some cases, the larger the value the better. A disk cache size of 512K is recommended; improved performance gains under Mach^{Ten} tend to diminish beyond this size. Much of the improvement seems to come when the system is manipulating folders or directories with large numbers of entries, as with the */usr/bin* and */etc* directories under Mach^{Ten}.

- **Turn off file sharing**

  Disable the file sharing option located in the Sharing Setup Control Panel.

- **Close the Control Panels after use**

  Do not keep the Mach^{Ten} scheduling priority slide bar or any Control Panel on the desktop. An open Control Panel consumes valuable CPU cycles.[†]

- **Adjust your search path**

  When running Mach^{Ten} in the default configuration, the following directories are searched (in the order listed) when a Mach^{Ten} command is executed:

  ```
  /usr/bin
  /bin
  /usr/X11/bin
  /usr/libexec
  /sbin
  /usr/sbin
  ```

  Adjust your search path in the *.login* or *.profile* file in your *home* directory.

- **Increase UFS & and NFS buffers**

  Increase the number of UFS and NFS buffers in the Mach^{Ten} Memory Control Panel. Increasing the number of buffers to 32 or 64 can significantly improve performance by increasing the frequency of file system cache hits.

2.2.1 System Optimization Guidelines

The System Optimization Guidelines Table which follows will help you customize Mach^{Ten} systems for particular environments. A little system attention will go a long way to give you a smoothly running system.

[†] Refer to section "3.0 The Mach^{Ten} Control Panel" for a more details on using the slide bar.

It is important to remember that Mach^{Ten} relies solely on physical memory. However, if you monitor your system resources and are careful not to overload your system with unneeded *daemons*, you will have good results with Mach^{Ten}, even on small memory machines.

Table 1: System Optimization Guidelines

Disk Space Limitations A system with 100MB or less of hard disk will probably require careful attention to "tidiness" of the available disk space.	• By default, the */etc/rc* file will preserve the editor files, and then automatically remove files in */tmp* on each restart. (Note that subdirectories in */tmp* are not purged.) • A *cron(8)* entry may be created to purge old 'preserved' editor files from */var/preserve*. • If a server system is available with more disk space, always use NFS. • Use *df(1)* to determine the capacity and available space of each mounted file system.
Memory Space Limitations Since Mach^{Ten} does not support paging, nor does it swap processes from memory to disk, all of the processes (blocked, sleeping and running) consume real memory. When the memory is filled up, no new processes can be started. Process instantiation (*execve(2)*) will fail with an ENOMEM (Error: not enough memory).	• Run only the daemon processes that are needed. If you do not need the services provided by a daemon, you can disable it via *rc.conf(5)* (see opposite). • Use *ps(1)* to find out how much memory each process is consuming. • If available memory is filled, terminate unwanted processes with *kill(1)*. This will free up the memory those processes were consuming. • Keep the maximum memory that you are consuming below a high water mark to avoid memory fragmentation problems. (The probability of memory fragmentation increases as the percentage of free space decreases.) • If you want to run many Macintosh applications at the same time, it is possible, under Finder, to reduce each Macintosh application's memory allotment: Select the Application; select Get Info from the File menu; double-click to select the number in the box labeled "Application Memory Size"; click on the Close box.
Daemon Processes NFS Server: This service is provided by three daemons: *portmap*, *mountd* and *nfsd*.	Run NFS daemons on NFS servers only. There is no advantage to running multiple *nfsd*'s under Mach^{Ten}; they only consume more memory.

syslogd	This is the system logger daemon. *Syslogd* reads and forwards system messages to the specified log files and / or users, depending on the priority of the message and the originating facility. A major reason for running this daemon is to log the results of *sendmail* activities. Run *syslogd* while changing system configuration files for other daemons, or when running *sendmail*, but not otherwise.
sendmail	*Sendmail* is a large application. It is required to receive network mail. It is not required to send network mail. If you are short on memory, you may want to *rlogin* to another machine or use a POP client to read your mail.
inetd	*Inetd* is only needed for incoming or loopback connections. If a machine is not providing *telnetd*, *rlogind*, *ftpd*, and other similar services, *inetd* does not need to be run. Note that *telnet*, *rlogin* and *ftp* clients (for outgoing connections) can still be run on demand.
Router Service	No daemons are necessary to provide IP forwarding, although you may choose to run *routed.* A system that is forwarding IP packets may allocate additional *mbufs* dynamically, but these allocations are small compared to the size of most programs.
Serial Ports	*Getty* is a process like any other, so an unused serial line with a waiting *getty* is consuming memory. Ttys can be turned on and off when needed (see *init(8)*).
wind The default *.profile* and *.login* files provided automatically start up the window daemon each time a login occurs in the console. Note that the window daemon is started with an `exec /usr/libexec/wind` to free up the space of the invoking call.	*Wind(8)* is the window daemon controlling the Mach^{Ten} window environment. It is not necessary to use the window daemon since Mach^{Ten} also supports a simple console device. However, the added value of the window environment is almost always worth the extra memory it consumes.
lpd	The printing service is provided by *lpd(8).* Systems that do not require printing services need not run this daemon. Mach^{Ten} will automatically start this daemon when a print command is invoked.
httpd	Run only if providing a World-Wide Web service.

2.2.2 How Can You Tell When a System is Approaching Its Limits?

A system that is running near full memory capacity will begin to slow down. Output to the console (or any of the windows) may be noticeably slower. Programs which request more memory may fail or simply block and wait until memory is available. The C shell will return "Not enough memory" when attempting to invoke a new program.

Remote machines which attempt to *rcp(1)*, *ftp(1)*, or *telnet(1)* to a loaded machine may see "Protocol error, server closed connection" or "/bin/csh: No memory" depending on how far the server side of the connection was able to progress before memory was exhausted.

2.3 Reinstallation

Occasionally you may want to reinstall Mach^{Ten}. Your current system may become damaged, or you may need to install a system upgrade. Perform the following steps before reinstalling Mach^{Ten}:

Step 1 — Preserve Personalized Files and Files That Have Been Changed

- Quit Mach^{Ten}.
- Create a folder named *Preserve* in the top level of your hard disk.
- Drag any configuration files you have changed to the *Preserve* folder. This includes many of the files located in */etc*, such as *fstab, group, hosts, inetd.conf, sendmail.cf* and *passwd*. It may be best to save all of */etc* and review the files after installation is complete. Private user files in the */home* directory should be saved. Note that the *.cshrc*, *.login*, *.mailrc* and *.profile* files for the *root* account (*/home/root*) may be updated in subsequent releases.
- Drag any other files you have changed in */dev*, */src* or */usr*, and/or binaries you have built in */usr/local* to the *Preserve* folder.
- Visit the *var* folder and determine whether or not to preserve the administrative log files, queued mail messages, print requests and accounting files contained in the following sub-folders *adm*, *log*, *spool*, *tmp* and *preserve*. If in doubt, preserve the file(s) and take a closer look after the installation is complete.

Step 2 — Remove Outdated Mach^{Ten} System Files From Your Hard Disk

- From the top-level folder, drag the following files/folders to the Trash: *bin, dev, etc, mnt, tmp, volume, usr, var* and *MachTen*.
- Empty the Trash.

Step 3 — Install the New Mach^Ten System

Step 4 — Restore the Saved Configuration Installation Parameters

- Before launching Mach^Ten, open the Mach^Ten Control Panel and visit each configuration screen. Verify that your installation parameters are correct. Press the "Save" button, even if no changes were made.

Step 5 — Launch Mach^Ten by Opening the Mach^Ten Application

Step 6 — Restore the Remaining Saved Configuration Files

- When the Mach^Ten installation has completed, compare any customized configuration files from the *Preserve* folder to the corresponding files in the */etc* and */home* folders.

NOTE: If you have not explicitly made changes to a configuration file, do not restore it. Configuration files sometimes change in new releases of Mach^Ten and restoring old configuration files may make the new release behave unexpectedly.

2.3.1 Reconfiguration

If your network configuration changes, or if you installed Mach^Ten before you were attached to a network, numerous files must be updated. This may be done automatically by using the Mach^Ten Control Panel or manually by updating the necessary files by hand. Some network service changes, such as disabling *ftpd* or enabling *fingerd,* will require manual intervention as noted below. Where possible, the automated approach to network reconfiguration is strongly recommended.

2.3.1.1 Automated Reconfiguration

In automated configuration, the following files are updated according to the information entered on the "Networking Configuration" screen of the Mach^Ten Control Panel:

```
/etc/rc.conf            /etc/hosts
/etc/resolv.conf        /usr/etc/zoneinfo/localtime
/etc/sendmail.cf        /usr/lib/sendmail
/etc/sendmail.cw
```

Take the following actions:

- Quit Mach^{Ten}
- Choose "Control Panel" from the Apple menu and open "MachTen Controls". Press the "Networking" button, make the appropriate changes, and press the "Save" button.
- Launch Mach^{Ten}

  During the normal boot process, updated configuration parameter values will be noted and applied. Comments similar to the following will be displayed:

```
      # Applying Installation configuration parameter settings..
 Creating directory "/etc/PRE.Aug_20_13:54:01" to preserve original copies
  Setting timezone to US/Pacific
  Updating /etc/hosts
  Creating /etc/sendmail.cw
  Updating /etc/sendmail.cf
  Creating /etc/resolv.conf
  Updating /etc/rc.conf

       # "/etc/PRE.Aug_20_13:54:01" may be removed when the original copies
       # it contains are no longer of interest for review or recovery.
```

2.3.1.2 Manual Reconfiguration

The following table lists the files you must edit for the indicated configuration change:

- domain name change
```
/etc/hosts
/etc/resolv.conf
/etc/sendmail.cf
```
- host name change
```
/etc/hosts
/etc/rc.conf       (if local host change)
/etc/sendmail.cw       (if local host change)
```
- IP address change
```
/etc/hosts
```
- network service change; for example, disabling *ftpd*
```
/etc/inetd.conf
```

- NFS server change
  ```
  /etc/fstab
  /etc/hosts
  ```
- route change
  ```
  /etc/hosts
  /etc/rc.conf
  ```
- local service change; for example starting *httpd* at Mach^{Ten} boot up
  ```
  /etc/rc.local
  ```

If you are changing the local host name and / or address, be sure to reboot Mach^{Ten} after updating the appropriate files in order to put them into effect.

2.3.2 Accessing Mach^{Ten} Files Directly From the CD-ROM

Several Mach^{Ten} files may be accessed directly from the CD-ROM, rather than copying them to your hard disk. Although CD-ROMs typically have slower access times than hard disks, this configuration significantly reduces the hard disk footprint necessary to take full advantage of the Mach^{Ten} distribution.

In order to access files directly from the CD, the CD must be online (visible on the desktop) and be mounted under Mach^{Ten}. The default */etc/fstab* file will mount the Mach^{Ten} CD on the path */CDROM* if the CD is online when Mach^{Ten} is started or if the CD is inserted when Mach^{Ten} is running.

Since the mechanism to access files directly from the CD-ROM uses symbolic links, the CD may be ejected once the appropriate files have been accessed, and re-inserted when the files are needed again.

2.3.3 Accessing Online Man Pages Directly From the CD-ROM

To access the online man pages from the CD, a symbolic link to the man page directory on the CD must be created. As *root*, execute the following commands:

```
# cd /usr/share
# mv man man-
# ln -s "/CDROM/Man Pages/tty" man
```

Note that subsequent attempts to install additional man pages from other applications you port to Mach^{Ten} will fail, as the CD-ROM is a read only disk.

2.3.4 Accessing Mach^{Ten} Sources Directly From the CD-ROM

To access the source files from the CD, a symbolic link to the source directory on the CD must be created. As *root*, execute the following commands:

```
# cd /base
# mv src src-
# ln -s /CDROM/source src
```

Note that the target directory */base/macppc* is still on the hard disk, so the objects for the BSD4.4 sources can be built even though the sources are on the CD. Obviously you cannot modify these programs by changing the source without copying them from the CD to your hard disk.

2.3.5 Accessing Mach^{Ten} X Window Software Directly From the CD-ROM

To access the Mach^{Ten} X Window software from the CD, a symbolic link to the X11R5 directory on the CD must be created. As *root*, execute the following commands:

```
# cd /usr
# ln -s "/CDROM/X Window Software/X11R5" X11R5
```

Obviously you cannot modify the X clients by changing the source without first copying them from the CD to your hard disk.

2.4 Troubleshooting

The Mach^{Ten} installation automatically configures the UNIX networking software and automatically adjusts the Macintosh system environment to ensure a properly running system. This section will help you solve problems that may arise as you work with Mach^{Ten}.

Symptom	Solution
While installing and copying files from the CD-ROM to the hard disk, a dialog box appears saying that a particular file cannot be copied and should be removed from the hard disk.	Temporarily remove or disable your INITs, particularly any anti-virus INITs. Retry the installation.

Symptom	Solution
Launching Mach^{Ten} causes a system "bomb".	There may be a Control Panel or Extension conflict. Try temporarily disabling all INITs except the MachTen INITs to determine if you have a conflict. • Make two new folders in your System Folder called *Control Panels check* and *Extensions check.* • "Select All" in the Control Panels folder, and then hold the shift key down and click "MachTen Controls". Drag the selected files into *Control Panels check* folder. • "Select All" in the Control Panels folder, and then hold the shift key down and click "MachTen NFS" and "Foreign File Access". Drag the selected files into *Extensions check* folder. • Restart your Macintosh. • Run Mach^{Ten}. If Mach^{Ten} successfully starts up, then a Control Panel or Extension in your system is incompatible with Mach^{Ten}. You can trace down the culprit by restoring and disabling your Control Panels and Extensions by halves until there is one remaining.
Mach^{Ten} seems to slow down.	Do a *ps -alx* and observe what processes are running. Use *kill(1)* to terminate unwanted processes. Read section "2.2 Performance Tuning — Optimizing Mach^{Ten}" for system optimization hints. If using Desktop NFS, make sure "Calculate Folder Sizes" is turned off in the Views Control Panel.
You notice loss of data due to file truncation.	Due to Mach^{Ten} file caching, Mach^{Ten} may not return an error when writing to a file system that is full. Check the free space on your file system before continuing.
You experience a power failure.	Mach^{Ten} must be restarted after a power failure. Traditional UNIX file systems are sensitive to power failures. Since Mach^{Ten} uses the Macintosh File System exclusively, it is not necessary to check the integrity of the file system.
Launching Mach^{Ten} causes the Control Panel Warning Dialog Box to appear.	Open the Mach^{Ten} Controls control panel. Configure and save the General, Memory and Networking Screens.

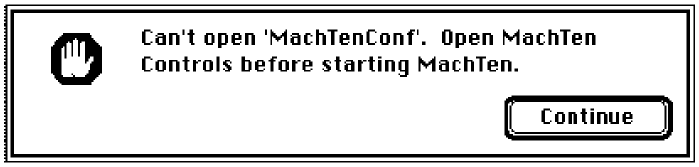
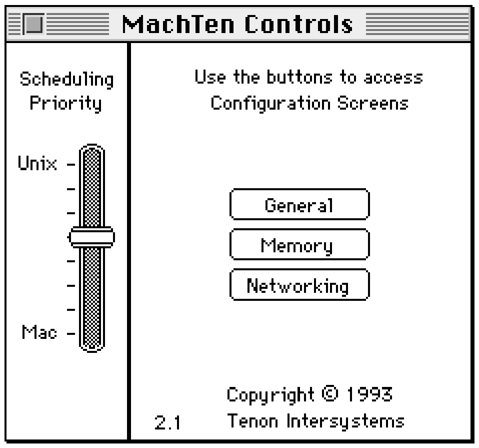

Can't open 'MachTenConf'. Open MachTen Controls before starting MachTen.

Figure 6. Control Panel Warning Dialog Box

3.0 The Mach^{Ten} Control Panel

The Mach^{Ten} Control Panel is used to configure certain parameters in your Mach^{Ten} system. It is normally opened once prior to launching Mach^{Ten} for the first time, and subsequently if you want to change any Mach^{Ten} operating parameters.

To view the Mach^{Ten} Control Panel, choose "Control Panels" from the Apple menu and double-click on "MachTen Controls". The Control Panel consists of the Scheduling Priority slide bar and three configuration buttons. With the exception of Scheduling Priority slide bar adjustments, all changes made in the Mach^{Ten} Control Panel require you to restart Mach^{Ten} in order for the modifications to take effect.


MachTen Controls

Use the buttons to access Configuration Screens

Copyright © 1993
Tenon Intersystems

Figure 7. Mach^{Ten} Control Panel

3.1 Scheduling Priority Slide Bar

On a busy system where both Mach^{Ten} and Macintosh programs are running simultaneously, the Scheduling Priority slide bar allows you to control the relative CPU time spent running UNIX programs versus running Macintosh programs. A change of priority goes into effect immediately, allowing you to tune your system depending on your personal workload priority. For example, if you are compiling UNIX programs using Mach^{Ten}'s "C" compiler (*cc*) while printing documents from a Macintosh application, the compiler will run relatively faster if you push the slide bar further in the UNIX direction. For optimal performance, the Mach^{Ten} Control Panel should be closed after making any scheduling priority adjustments.

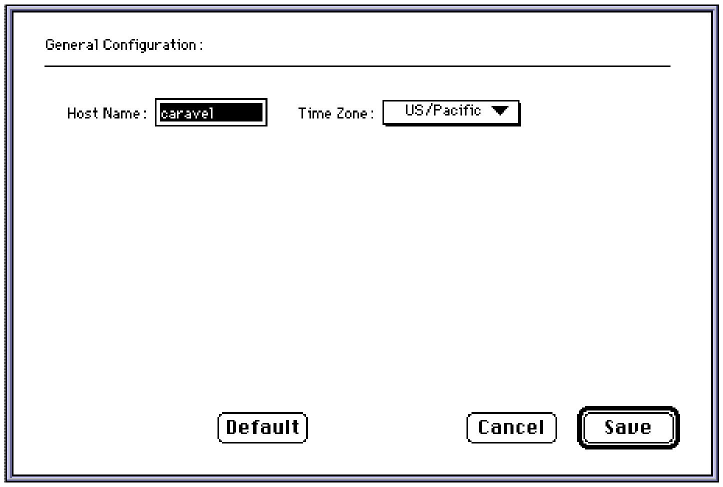
The Mach^{Ten} scheduler is designed to optimize performance independent of the slide bar position. Therefore, when no other Macintosh applications are running, the slide bar position will have no effect on the performance of Mach^{Ten}. Similarly, when no Mach^{Ten} processes are active, the impact of Mach^{Ten} on Macintosh applications will be minimal, regardless of the slide bar setting.

3.2 Configuration Screens

There are some basic configuration parameters that must be set in order for Mach^{Ten} to operate properly. These include general Mach^{Ten} system information, settings that affect Mach^{Ten} memory utilization, and addressing and server information if participating on a network.

The configuration screens are activated by clicking on the appropriate button in the control panel. Each screen contains edit fields and check boxes for specifying configuration values. The “Default” button in each configuration screen will reset the entries in the screen to installation default values. After entering your configuration values, click on the “Save” button to confirm your entry, or on the “Cancel” button to leave the values unchanged. Changed values will take effect the next time you start up Mach^{Ten}.

3.2.1 General Configuration Screen


General Configuration:

Host Name: caravel

Time Zone: US/Pacific

Figure 8. “General Configuration” Screen

3.2.1.1 Host Name

This is the name given to your Macintosh. The host name entered will be used in conjunction with addressing information from the Network Configuration screen (described below) to uniquely identify your system on an internet network. The default host name will be taken from the “Macintosh Name” field in the Network Identity section of the Sharing Setup Control Panel. Under Mach^{Ten}, blanks and special characters other than periods and underscores are not allowed in a host name. Special characters found in the default host name are converted to underscores.

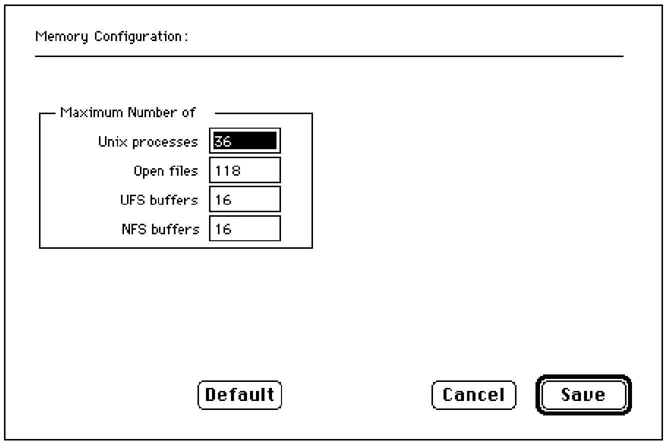
3.2.1.2 Time Zone

This is a pop-up menu for choosing the appropriate time zone for your system. Many UNIX systems set the system clock to Greenwich Mean Time (GMT) and automatically compensate for Daylight Savings Time. In order to integrate with time as seen on the Macintosh, Mach^{Ten} uses the time as set in the Macintosh General Control Panel, typically local time. The time zone information helps Mach^{Ten} convert between local time and GMT. Users are responsible for adjusting the General Control Panel clock by hand when changing to and from Daylight Savings Time.

If available, the default time zone is extracted from the setting in the Macintosh World Map Control Panel.

3.2.2 Memory Configuration Screen

Mach^{Ten} uses maximum parameter values in this screen to dynamically allocate UNIX kernel memory resources at boot time. As your system requirements grow, these values can be adjusted upward accordingly.


Memory Configuration:

Maximum Number of

Unix processes	36
Open files	118
UFS buffers	16
NFS buffers	16

Figure 9. "Memory Configuration" Screen

3.2.2.1 Maximum Number Parameters

UNIX kernel parameters are presented as maximum allowable values:

UNIX processes. This is the maximum number of processes (shells, programs, daemons, etc.) that can be run at one time. The default value of 36 processes should be adequate for single user Macintoshes. If you have multiple users logged in to your system over the network, you may have to increase this parameter.

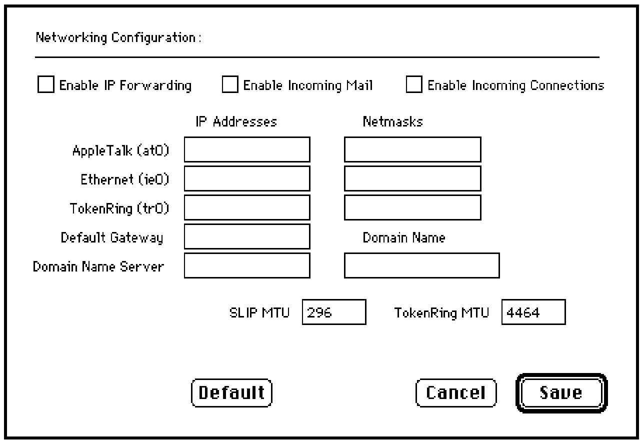
Open files. This entry specifies the number of files Mach^{Ten} can have open and in memory at once. The default number of open files is 118.

UFS buffers. This entry is the number of 8 kilobyte UNIX File System (UFS) buffers available for buffering file transfer to and from the local disk. Increasing this number from the default 16 buffers will speed up your system at the expense of memory usage.

NFS buffers. This entry is the number of 8 kilobyte Network File System (NFS) network buffers available for NFS access. Increasing this number from the default 16 buffers will speed up your system at the expense of memory usage.

3.2.3 Networking Configuration Screen

Mach^{Ten} allows for a multi-homed internet networking environment — permitting multiple independent networks to be connected to a single system. Mach^{Ten} supports several physical interface types, including Ethernet, SLIP, TokenRing and LocalTalk, in addition to a logical local loopback interface. Mach^{Ten} also supports multiple interfaces of the same type, including up to four Ethernets, three SLIP and / or PPP lines[†], four TokenRings, and a single LocalTalk.


Networking Configuration :

☐ Enable IP Forwarding ☐ Enable Incoming Mail ☐ Enable Incoming Connections

	IP Addresses	Netmasks
AppleTalk (at0)		
Ethernet (ie0)		
TokenRing (tr0)		
Default Gateway		Domain Name
Domain Name Server		

SLIP MTU 296 TokenRing MTU 4464

Figure 10. "Networking Configuration" Screen

Mach^{Ten} will extract all default network parameters from the MacTCP Control Panel, if available.

(See section "9.0 Networking with Mach^{Ten}" for more details on internetworking topics introduced by the Networking Configuration screen.)

[†] The default number of serial interfaces can be increased to satisfy custom serial interface requirements. See the *Configuration Resources* technical note in the */pub/tech_notes* directory on *ftp.tenon.com*.

3.2.3.1 Enable IP Forwarding

Check to enable forwarding of Internet Protocol packets. IP packet forwarding enables the system to act as a router between LocalTalk and Ethernet or other interfaces.

3.2.3.2 Enable Incoming Mail

If you want to receive Internet network mail messages directly on your system, check this box. When checked, the UNIX *sendmail* daemon is started automatically when Mach^{Ten} is launched. If you use any other mail program (e.g., QuickMail, ccMail, Eudora), leave this box unchecked. By default, *sendmail* is disabled.

NOTE: There is no need to enable *sendmail* to send electronic mail to the network or to send and receive local Mach^{Ten} mail among user accounts on your Mach^{Ten} Macintosh.

3.2.3.3 Enable Incoming Connections

Check to accept connection requests for Internet network services. When checked, Mach^{Ten} will start *inetd*, the Internet "super server", automatically upon Mach^{Ten} launch. *inetd* listens for connections on well known Internet "ports" such as Telnet and FTP. With *inetd* disabled, connection requests for these services will be denied. By default, *inetd* is enabled.

NOTE: Unlike *sendmail*, *inetd* must be enabled to accept local loopback connections on the well known Internet ports.

3.2.3.4 Internet (IP) Addresses

The Networking Configuration screen is a convenient way to configure the primary interface on AppleTalk, Ethernet and TokenRing for internet networking. If you have multiple interfaces of the same type, refer to section "9.0 Networking with Mach^{Ten}". Also detailed are instructions on how to configure PPP or SLIP interfaces; see the appropriate section, "9.13.8 Using PPP" or "9.13.9 Using SLIP". Each configured interface (described below) must be assigned a unique internet address in standard Internet "dot" (decimal) format. The two-letter description in parentheses is a short version of the interface name as used by Mach^{Ten}. It is followed by a zero, indicating the primary interface.

AppleTalk (at0). If your Macintosh is connected to a LocalTalk network or your MacTCP Control Panel is configured for AppleTalk over Ethernet ("EtherTalk") or for AppleTalk over TokenRing ("TokenTalk"), assign an Internet address to the AppleTalk interface. The default value, if shown, comes from your MacTCP configuration. You will be able to use Apple networking facilities (printers, file sharing, etc.) without assigning an Internet address to your LocalTalk interface; however, if you wish to have IP connectivity on AppleTalk, it must have an Internet address. (This is useful if, for example, users with NCSA Telnet on AppleTalk want to connect to your Mach^{Ten} Macintosh directly over AppleTalk.) If a MacIP gateway and dynamic addressing are used on your network, then enter an IP address of "0.0.0.0".[†] (Unless you plan to use your Mach^{Ten} networking code solely for Telnet access to some other system, it is recommended that you do not use dynamic Internet addressing.)

In a system with more than one type of networking interface, the host name assigned by Mach^{Ten} to the AppleTalk interface will be "*host*_a", where *host* is the host name selected in The General Configuration Screen. If this interface is the only configured interface, the host name associated with the IP address entered will simply be *host*.

Users running EtherTalk or TokenTalk on their system may elect to configure the corresponding Ethernet or TokenRing interface under Mach^{Ten} *in addition to* the AppleTalk interface. This requires the assignment of a unique IP address (that is, different than the address assigned to AppleTalk) to the Ethernet or TokenRing interface described below. In this configuration on Ethernet, users would have direct access to other non-Macintosh internet hosts from Mach^{Ten} without the overhead of EtherTalk packet conversion by an AppleTalk to Internet gateway while still enjoying connectivity with AppleTalk hosts via EtherTalk.

In effect, two logically distinct network nodes are created on your Macintosh which share a common Ethernet cable to send and receive packets. The host name on EtherTalk assigned by Mach^{Ten} and corresponding to the AppleTalk IP address entered would be "*host*_a" and the Ethernet host name corresponding to the Ethernet IP address entered would be "*host*" with an alias of "*host*_e", where *host* is the host name selected in The General Configuration Screen of the Mach^{Ten} Control Panel.

Ethernet (ie0). If your Macintosh is connected to an Ethernet, this field represents the assigned Internet address of the Ethernet network interface. The IP address from the MacTCP Control Panel will appear by default if MacTCP has been configured for communication over Ethernet.

In a multiple interface configuration, the host name assigned by Mach^{Ten} to this interface will be "*host*", where *host* is the host name selected in the General Configuration screen. The host name alias "*host*_e" will also be assigned to this interface.

TokenRing (tr0). If Your Macintosh is connected to a TokenRing, this field represents the assigned Internet address of the TokenRing network interface. The IP address from the MacTCP Control Panel will appear by default if MacTCP has been configured for communication over TokenRing.

In a multiple interface configuration, the host name assigned by Mach^{Ten} for this interface will be "*host*_t", where *host* is the host name selected in the General Configuration screen. If this interface is the only configured interface, the host name associated with the IP address entered will simply be *host*.

[†] By default, Mach^{Ten} scans the locally connected AppleTalk zone for an IP gateway that will provide the IP address. Mach^{Ten} may be configured to scan multiple zones to locate a MacIP gateway during dynamic IP address assignment. See the *Configuration Resources* technical note in the */pub/tech_notes* directory on *ftp.tenon.com*.

Default Gateway. This field represents the internet address in standard Internet "dot" format of the default gateway system on your network, and defaults to the *gateway* field in the MacTCP Control Panel. All access to systems not on your network will be made through the default gateway.

Domain Name Server and Domain Name. The Domain Name Server field represents the internet address in Internet "dot" format of the network domain name server. The domain name is an ASCII string (for example, *tenon.com*) representing your network domain name. Both entries default to the "Default" Domain Name Server IP Address and Domain in the MacTCP Control Panel.

3.2.3.5 Netmasks

If subnetworking is implemented on your network, the Netmask fields are used to derive the network and host (node) portions of the Internet address for a given network interface.

Netmasks are entered either in standard Internet "dot" (decimal) format or in hexadecimal format preceded by the characters '0x'. The default value is taken from the MacTCP Control Panel. If subnetworking is not employed on any of your networks, the Netmask field for each configured interface should be left blank. In this case, Mach^{Ten} will determine the correct netmask based on the interface IP address.

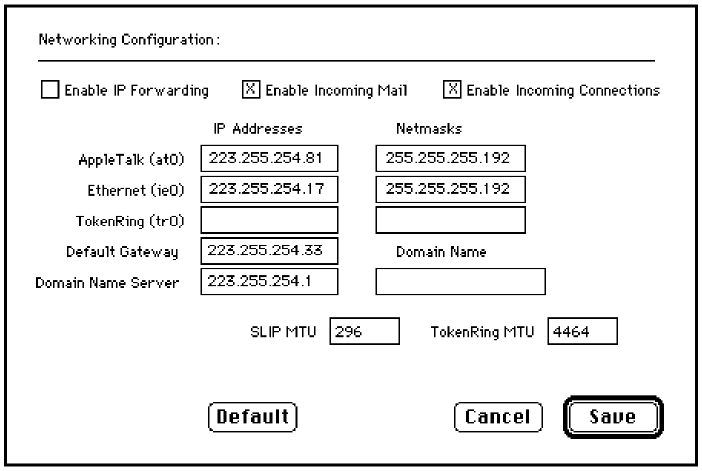
3.2.3.6 Maximum Transmission Unit Settings

The maximum transmission unit (MTU) sets the maximum output packet size for connections on a particular network interface. Mach^{Ten} allows the adjustment of the MTU for connections over SLIP lines and on TokenRing networks. For the LocalTalk, Ethernet and local loopback interfaces, the MTU is fixed at 576, 1500 and 1536 bytes, respectively. The MTU for PPP interfaces defaults to 1500 bytes and is negotiated by the PPP client and server during connection establishment. The MTU setting will be applied to all interfaces configured in systems with multiple interfaces of the same type.

SLIP MTU. This parameter sets the maximum output packet size for SLIP connections. The default value is 296 bytes. Increasing this value will improve network performance at the expense of interactive response time.

TokenRing MTU. This parameter sets the maximum output packet size for the TokenRing interface, if configured. The default value is 4464 bytes.

Example Network Configuration. The following figure illustrates a fully configured networking configuration screen for a Macintosh with both an Ethernet and an AppleTalk interface in the *tenon.com* domain. Both interfaces are part of a subnetworking environment. Packets to non-*tenon.com* hosts will be sent to an IP gateway on Ethernet. For host name resolution, this Macintosh will use a domain name server on Ethernet.


Networking Configuration:

☐ Enable IP Forwarding [✓ Enable Incoming Mail] [✓ Enable Incoming Connections]

	IP Addresses	Netmasks
AppleTalk (at0)	223.255.254.81	255.255.255.192
Ethernet (ie0)	223.255.254.17	255.255.255.192
TokenRing (tr0)		
Default Gateway	223.255.254.33	
Domain Name Server	223.255.254.1	

Domain Name:

SLIP MTU 296 TokenRing MTU 4464

Figure 11. Example Network Configuration

4.0 Launching Mach^Ten

Mach^Ten consists of a Macintosh application, called *MachTen*, and a collection of shared libraries that contains the bulk of the Mach^Ten system software — the UNIX *kernel.* When the system is installed, these files are stored in a top-level *MachTen* folder and a *MachTen Libraries* sub-folder.

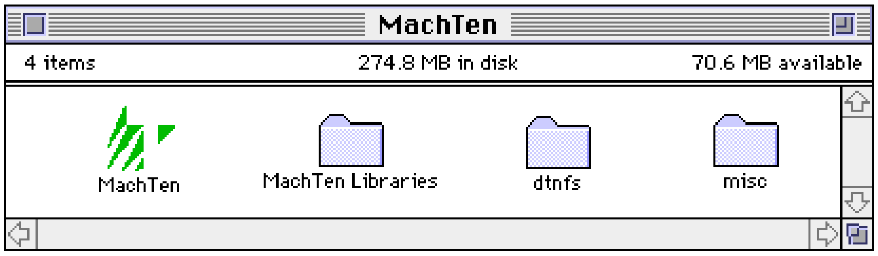


Figure 12. Mach^Ten Application Folder

When you double-click on the *MachTen* icon, Mach^Ten executes as a standard Macintosh application.

In the time between the click on the *MachTen* application until the login prompt appears, Mach^Ten initialization takes place (see Figure 13). The UNIX boot process is controlled by parameterized scripts which work in conjunction with three UNIX database files (*MachTen Prefs, MachTen Conf* and *Install.conf*) built by the Mach^Ten Control Panel.

The Mach^Ten application's function is to load the Mach^Ten libraries into memory and begin executing as a UNIX kernel. The UNIX kernel reads *MachTen Prefs* and configures some internal data tables accordingly. The kernel then starts the first UNIX process, *init(8).*

Init establishes a default environment for all subsequent processes. It sets certain environment variables and takes care of Macintosh time to GMT mappings. Once the default environment is set up, *init* starts a program which reads the */etc/rc* database and executes the commands in it. With input from the *rc.conf* configuration file generated from the Mach^Ten Control Panel settings, the network interfaces are activated. Also, several server programs, or *daemons*, are started (see Table 2). If successful in completing all of the commands in */etc/rc*, a login console is displayed.†

The daemons are described in detail in section 8 of the UNIX man pages. In Mach^Ten, the startup data files (Table 3) may be configured with a Macintosh word processor (before or after Mach^Ten is started), or via the Mach^Ten Control Panel. Tenon recommends using the Mach^Ten Control Panel whenever possible. Mach^Ten data files are described in detail in section 5 of the man pages.

† Mach^Ten may be configured to terminate the Macintosh Finder during the kernel boot up process. Running Mach^Ten without the Finder can increase Mach^Ten application performance and enhance the security of your Mach^Ten system. Refer to the *Configuration Resources* technical note in the */pub/tech_notes* directory on *ftp.tenon.com* for more information.

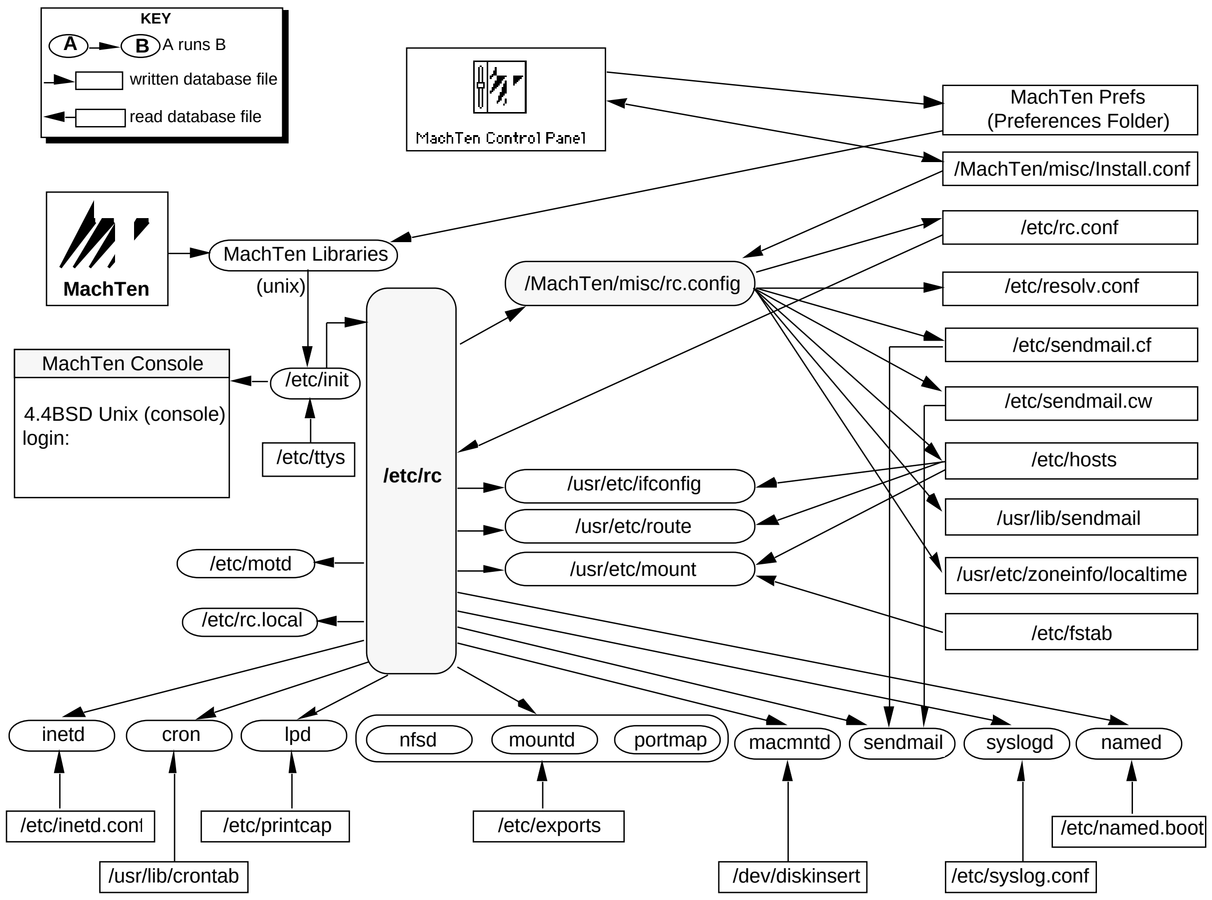


Figure 13. Mach^{Ten} Initialization

Table 2: Mach^{Ten} Daemons

rc	Main configuration file controlling startup and customization of UNIX. Reads *rc.conf* file created via Mach^{Ten} Control Panel and starts specified daemons.
inetd	Controls incoming network connections for these services: *ftp, rsh, rlogin, rexec, uucp, tcptraffic, pop, finger, tftp, ntalk, comsat, name, daytime, time, echo* and *discard*. Configured via */etc/inetd.conf* database.
cron	Controls starting of services or system tasks. Configured via */usr/lib/crontab* database.
lpd	Controls printing. Configured via */etc/printcap* database.
nfs, mountd, portmap	Control NFS service if system is exporting file system to other network clients. *mountd* uses */etc/exports* database to control which file systems are exported.
mactcpd	Replaces the Macintosh TCP service.
macmntd	Senses removable disk insertion (floppy, CD, etc.) and automatically mounts the volume.
sendmail	Accepts e-mail coming in from network connection. Configured via */etc/sendmail.cf* and / *etc/sendmail.cw* databases.
syslogd	Logs system activities or errors. Configured via *syslog.conf* database.
named	Provides name service. Maps host names to Internet addresses and vice versa. Configured via */etc/named.boot* database.
wind	Controls the Mach^{Ten} windows environment after log in.
httpd	Serves World-Wide Web (WWW) documents to internet Web browsers.

Table 3: Mach^{Ten} Data Files

/etc/ttys	Controls logins on the UNIX system terminals.
/etc/motd	Message of the day. Built by */etc/rc* process and used by login process each time user logs in.
/etc/rc.local	Invoked by */etc/rc* process. Should be used to configure site-specific daemons.
/MachTen/misc/rc.config	Invoked by */etc/rc* process. Reads *Install.conf* database built by Mach^{Ten} Control Panel and configures several system databases accordingly.
/etc/resolv.conf	Controls where client requests for name services are sent. Could be configured for a Name Server running on this system or another networked system.
/usr/etc/zoneinfo/localtime	Controls the time zone mappings to GMT.
/etc/rc.conf	Used by *rc*. Reflects parameters from the Networking and General Configuration Screens of the Mach^{Ten} Control Panel.
/etc/fstab	Database of file systems. Used by *mount(1).*
sendmail.cf	*Sendmail(8)* configuration file.
/etc/hosts	Host name-to-internet-address mapping.

4.1 The Mach^{Ten} Login Console

When Mach^{Ten} is first started it displays a command window and asks users to identify themselves with a user name and password. At this point, the desktop looks like a conventional Berkeley 4.4BSD UNIX system. (See Figure 14.)

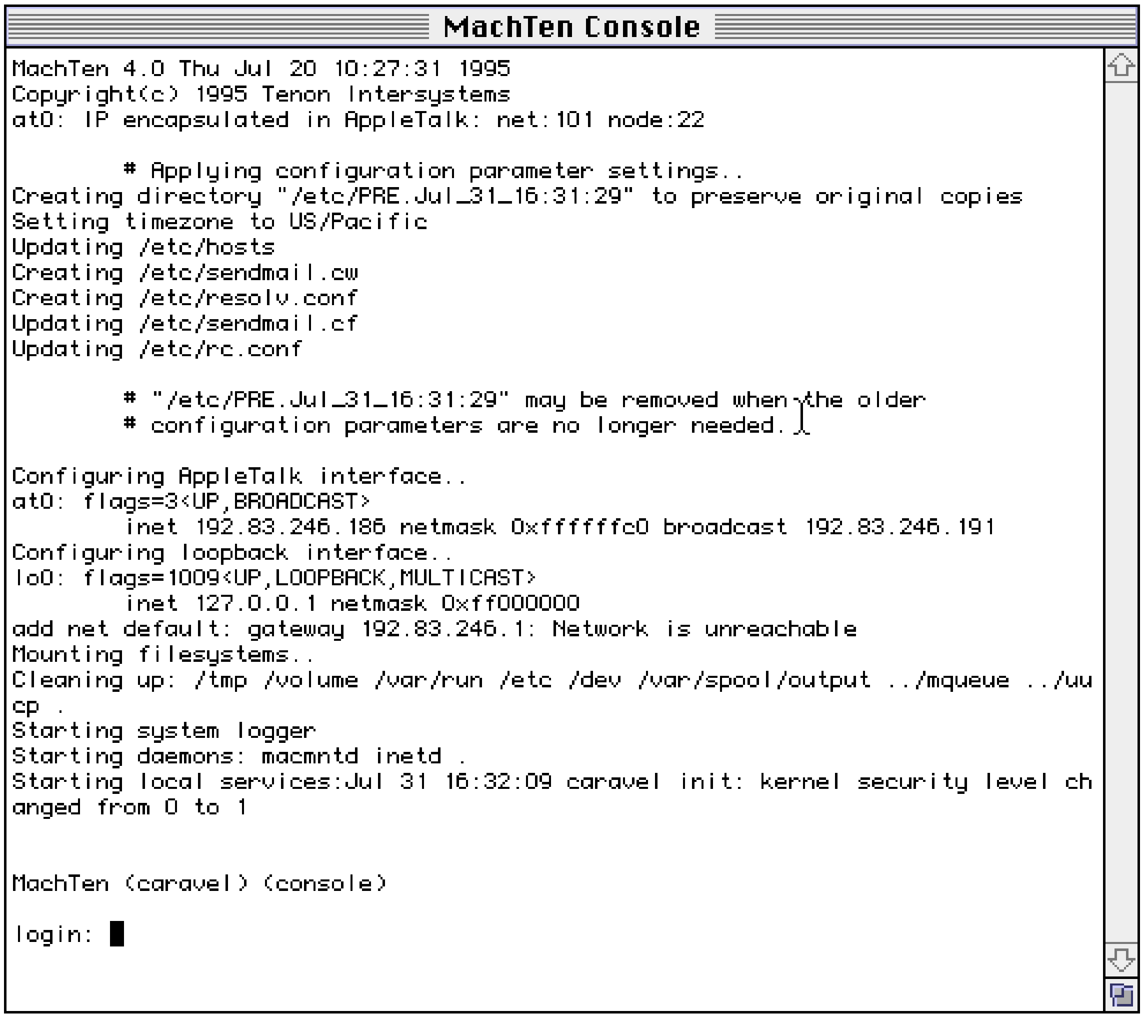
In the example in Figure 14, "caravel" is the "host name" that was entered into the Control Panel when Mach^{Ten} was installed. Log in as *root* (by typing "root" followed by a carriage return) and observe what happens. Be careful to use lowercase. Unlike the Macintosh, Mach^{Ten} is case sensitive. If you make a mistake you can use the Delete key to correct your error. When *root* logs in, the console window is automatically closed and a Mach^{Ten} "terminal window" is opened. See Figure 15.

After logging in, you enter a windows environment controlled by the *wind(8)* program. Mach^{Ten} windows let you create multiple windows, each representing a UNIX *shell* (the UNIX command interpreter). When you type commands in a Mach^{Ten} "terminal window", you are communicating with the shell. Mach^{Ten} provides both the Bourne shell (*sh(1)*) (the default) and the C shell (*csh(1)*). "Mach^{Ten} Windows" are described in detail in the next section.

Once you log in, you should immediately invoke the *passwd(1)* program and create a password for the *root* login. This is done by typing the following:

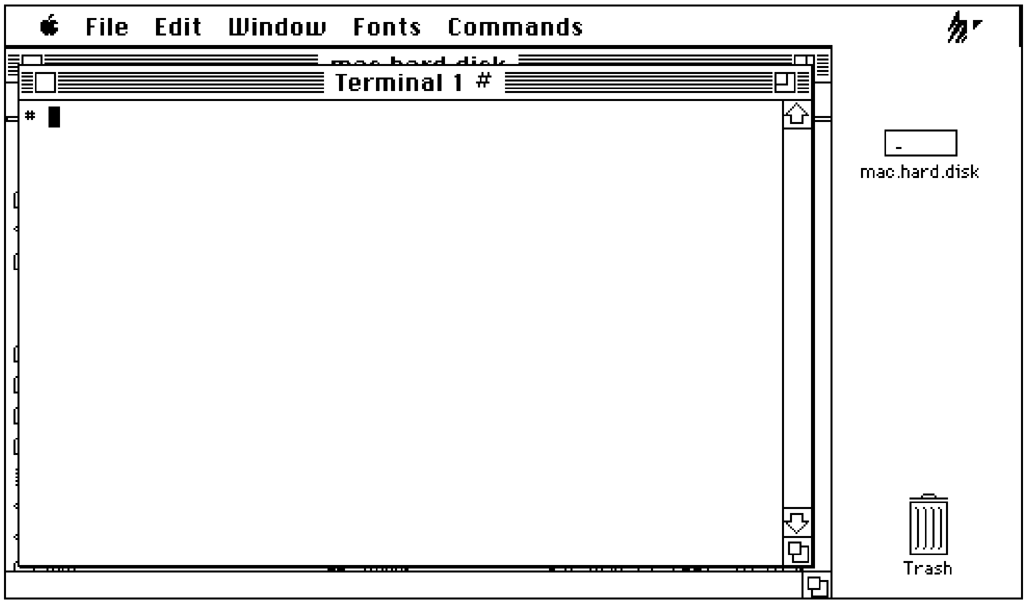
```
#passwd
New password: <new_root_password>
Retype new password: <new_root_password>
```

For more information refer to section "5.6.5 The Root Password".



```
MachTen 4.0 Thu Jul 20 10:27:31 1995
Copyright(c) 1995 Tenon Intersystems
at0: IP encapsulated in AppleTalk: net:101 node:22

        # Applying configuration parameter settings..
Creating directory "/etc/PRE.Jul_31_16:31:29" to preserve original copies
Setting timezone to US/Pacific
Updating /etc/hosts
Creating /etc/sendmail.cw
Creating /etc/resolv.conf
Updating /etc/sendmail.cf
Updating /etc/rc.conf

        # "/etc/PRE.Jul_31_16:31:29" may be removed when the older
        # configuration parameters are no longer needed.

Configuring AppleTalk interface..
at0: flags=3<UP,BROADCAST>
        inet 192.83.246.186 netmask 0xffffffc0 broadcast 192.83.246.191
Configuring loopback interface..
lo0: flags=1009<UP,LOOPBACK,MULTICAST>
        inet 127.0.0.1 netmask 0xff000000
add net default: gateway 192.83.246.1: Network is unreachable
Mounting filesystems..
Cleaning up: /tmp /volume /var/run /etc /dev /var/spool/output ../mqueue ../uu
cp .
Starting system logger
Starting daemons: macmntd inetd .
Starting local services:Jul 31 16:32:09 caravel init: kernel security level ch
anged from 0 to 1


MachTen (caravel) (console)

login: █
```

Figure 14. MachTen Login Console



```
# 
```

Figure 15. Mach^{Ten} Terminal Window

At this point you may wish to set up a user account for yourself. Refer to section "5.6.6 Setting Up User Accounts".

Once the system recognizes a user, it sets the current working directory, executes a series of commands to perform any automatic startup processing, checks to see if any electronic mail has arrived, and prints a prompt ($)[†] indicating that the system is waiting for input. To establish a new password, use the *passwd(1)* command, just as you did before as *root*.

4.2 Mach^{Ten} Windows

The Mach^{Ten} windows environment is a menu-oriented system enabling the user to create and control multiple command interpreter windows. It is created after you log in on the Login Console and is controlled by *wind(8)*, the Mach^{Ten} window daemon. The windows environment closely follows the Macintosh user interface standard. The windows of each Macintosh program are logically grouped together. All UNIX-style terminal windows are grouped together as Mach^{Ten} windows. Each window is a Macintosh "zoom window" with a title bar, scroll bar, zoom box, size box, and close box. The title bar of the active terminal window will be highlighted (see Figure 16).

[†] The '$' prompt is the default prompt for *sh(1)* users; for *csh(1)* users, the default prompt string includes your user name, host name and the '%' character in the form *username@host%*.

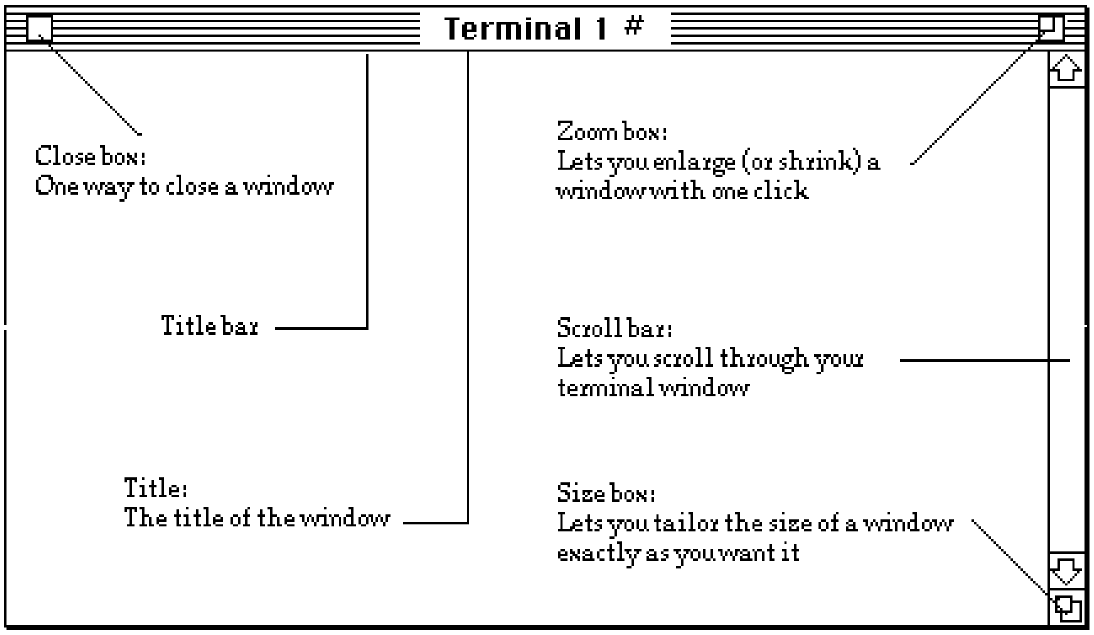

Terminal 1 #
Close box:
One way to close a window
Title bar
Title:
The title of the window
Zoom box:
Lets you enlarge (or shrink) a window with one click
Scroll bar:
Lets you scroll through your terminal window
Size box:
Lets you tailor the size of a window exactly as you want it

Figure 16. Mach^{Ten} Windows

Mach^{Ten} menus are located at the top of the screen and operate like traditional Macintosh menus.

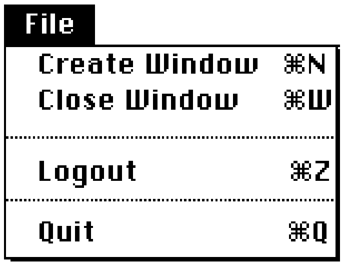
4.2.1 The Apple Menu

The Apple menu gives you access to About Mach^{Ten} and whatever desk accessories are installed on your Macintosh.

About MachTen	Information on how to contact Tenon

4.2.2 The File Menu

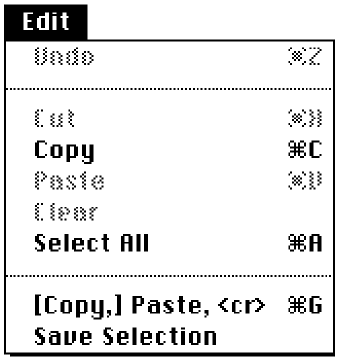
The File menu lets you create Mach^{Ten} windows and manipulate your window environment.



Create Window	Command-N	Create a new terminal window
Close Window	Command-W	Close the active terminal window
Logout	Command-Z	Quit windows and return to the Login Console
Quit	Command-Q	Quit Mach^{Ten}

4.2.3 The Edit Menu

The Edit menu gives you access to the Macintosh Clipboard. Text can be selected by moving the mouse cursor to the start / end of a selection and holding the button down while moving forward / backward to the end point of your selection, and then releasing the button. Selected text is displayed as white letters on a black background. Double-clicking the mouse selects the word where the mouse cursor is; triple-clicking selects the entire line.

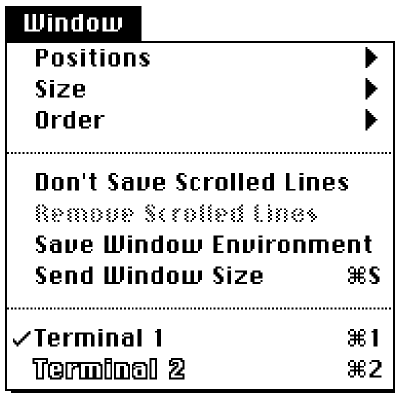


Undo	Command-Z	Undo is provided only to support desk accessories and is not active under Mach^{Ten}.

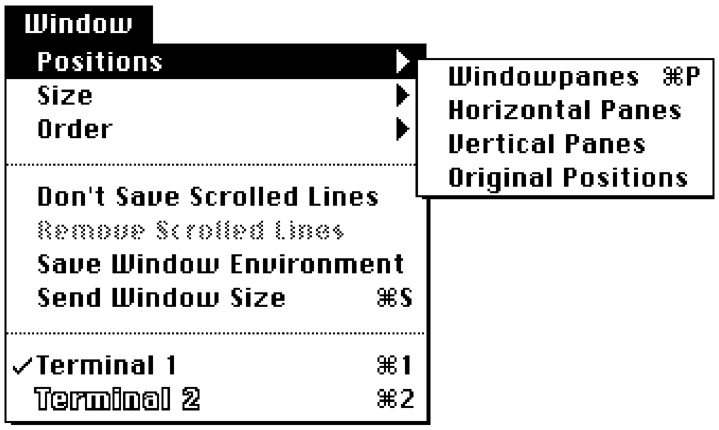
Cut	Command-X	Cut is provided only to support desk accessories. The concept of "cutting" from a UNIX "history" screen is not meaningful. Moreover, specific applications need built-in "cut" support. In Mach^{Ten}, traditional "cut-and-paste" functions become "copy-and-paste".
Copy	Command-C	Put a copy of the selection on the Clipboard.
Paste	Command-V	Put a copy of the contents of the Clipboard into the active window. The same item may be pasted many times until the next Copy command.
Clear		Clear is provided only to support desk accessories and is not active under Mach^{Ten}.
Select All	Command-A	Select all the text in the active window.
[Copy], Paste, <cr>	Command-G	Copy current selection to the Clipboard. The Clipboard is then provided as input to the active window, followed by a carriage return. This lets you select something from the screen and turn it into (the tail end of) a UNIX command.
Save Selection		Save the selected text to a file.

4.2.4 The Window Menu

The Window menu begins with three hierarchical menu items. To access these menus, slide the mouse cursor over the item and to the right. This will make their sub-menus visible. The other items in the Window menu are the names of existing terminal windows. The item with a check mark is the current console window. The active window is displayed in outline font. The keyboard shortcut for these windows is <command>-#, where # is the number of the terminal window.

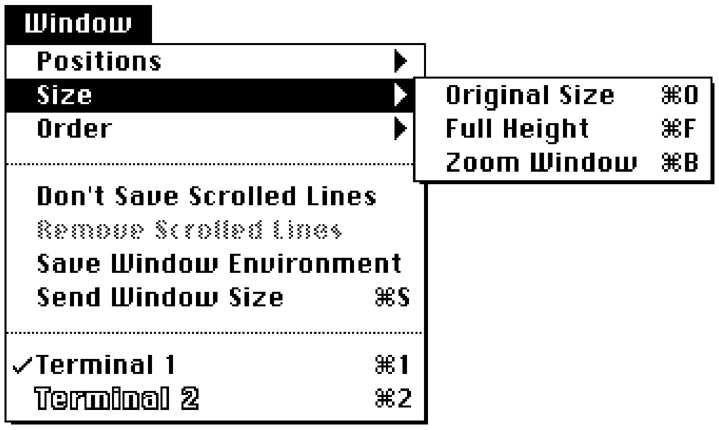


4.2.4.1 The Positions Sub-Menu



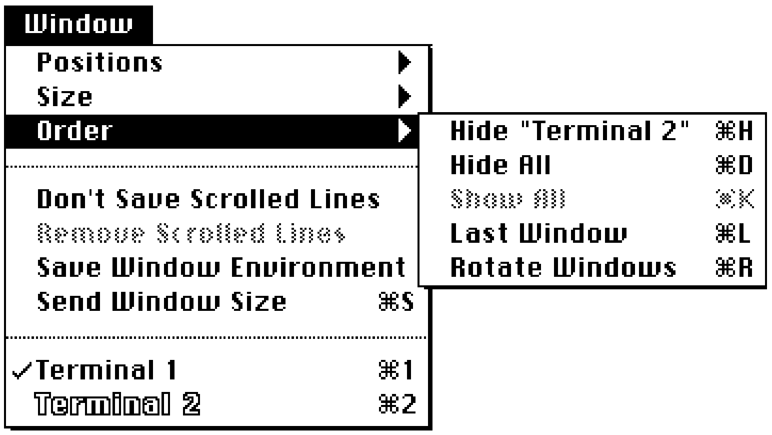
Windowpanes	Command-P	Position the windows so that they are all visible on the screen, creating windowpane effect.
Horizontal Panes		Windowpane effect with each window as wide as the complete screen.
Vertical Panes		Windowpane effect with each window as tall as the complete screen.
Original Positions		Undo the windowpane effect.

4.2.4.2 The Size Sub-Menu



Original Size	Command-O	Return the active window to the size it was when it was opened.
Full Height	Command-F	Make the active window as tall as will fit on the screen.
Zoom Window	Command-B	Make the active window as big as will fit on the screen. This is a toggle command — if the active window is full size, it will return to its previous size. The results of this command are identical to clicking in the zoom box of the active window.

4.2.4.3 The Order Sub-Menu

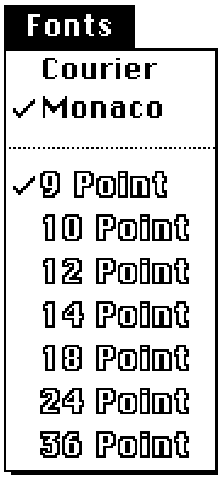


Hide Terminal #	Command-H	Make the active window invisible. A hidden window can be made visible using the Show All command (described below), or by selecting that window.
Hide All	Command-D	Make all windows invisible.
Show All	Command-K	Make all windows visible.
Last Window	Command-L	Make the previously active window the current active window.
Rotate Windows	Command-R	Make the current rear window (furthest from the front on the desktop) the front active window, and make the active window the second window.

Save Scrolled Lines / Don't Save Scrolled Lines		If "Save Scrolled Lines: is selected for a given window, the lines scrolled off the top of the window will be saved and the scroll bar will be activated. The window will stay in this state until "Don't Save Scrolled Lines" is selected. These two commands share one line in the Window menu; when "Save Scrolled" is enabled, the menu save "Don't Save", and vice versa.
Remove Scrolled Lines		If "Save Scrolled" is enabled, this will throw away any saved lines.
Save Window Environment		Save a description of the size and locations of the current Mach^{Ten} windows environment to a named file. Font types and sizes are also saved. This file is suitable to become the Mach^{Ten} window environment startup file.
Send Window Size	Command-S	Provide the string `stty rows n columns m` as input to the current active window, where `n` is the number of rows that will fit in that window using the current font type and size, and `m` is the number of columns. This is useful when accessing network-based UNIX systems that want to know the size of the terminal. An example would be using *telnet* to access a Sun workstation, and trying to run a screen-oriented program such as *vi* on the remote system. After connecting to the remote system, the *stty(1)* command will inform the remote system of the terminal size to use.

4.2.5 The Fonts Menu

The Fonts menu is created at startup time and contains all fixed-width fonts found in the current system, with point size choices of 9, 10, 12, 14, 18, 24, and 36. The current font size of the active window is checked, and the font sizes of “real” fonts are displayed in outline font. Real fonts are non-scaled and tend to look nicer for standard UNIX command interactions. Some commonly found fixed-width fonts are Courier and Monaco.



Font sizes can be changed to the next larger/smaller size by typing <command>-<up arrow> and <command>-<down arrow>.

NOTE: Variable width fonts in the system will not appear in the Mach^{Ten} Fonts Menu.

5.0 Mach^{Ten} Administration

This section deals with Mach^{Ten} system initialization, user accounts, file system maintenance, and virtual memory.

Mach^{Ten} requires considerably less "administration" than traditional UNIX systems. Since the underlying operating system is the Macintosh Operating System (MacOS), running Mach^{Ten} is, in general, no more complex than running a MacOS application. Mach^{Ten}, however, is a multiuser system with the concept of 'logins', 'passwords', and file permissions. Further, it contains sophisticated networking software. System administration for Mach^{Ten} includes establishing host and domain names for standard internet protocol interactions, as well as setting Mach^{Ten} up as a gateway, an NFS client or server, as a mail host, or as a UUCP participant.

Mach^{Ten} system administration can be performed by ordinary users. If Mach^{Ten} systems are integrated into a corporate or campus network, the network set-up tasks will no doubt be performed by the site network administrator. In an environment where there is already a UNIX administrator, Mach^{Ten} systems can easily be incorporated into the standard UNIX environment.

5.1 A Word About Man Pages

Mach^{Ten} supports a vast array of UNIX commands and utilities. All of the UNIX commands are documented in a set of "man" pages included with Mach^{Ten} (see section "2.1 Optional Installations"). The UNIX "man" pages are divided into numbered sections. Commands in this manual and all other Mach^{Ten} documents are referenced using these section numbers (e.g., *gawk(1)*).

The man page sections are:

Section No.	Section Title
1	User Commands
2	System Calls
3	Library Functions
4	Special Files
5	File Formats
6	Games
7	Public Files, Tables & Troff Macros
8	Maintenance Commands
9	X Window System Commands

5.2 Tailoring the Startup Environment

Changes made in the Mach^{Ten} Control Panel screens are recorded in the file */MachTen/misc/Install.conf+*. Whenever */etc/rc* sees changes in this file, it invokes the shell script */MachTen/misc/rc.config* to update several system files, including */etc/rc.conf*, to the new parameter values. */etc/rc* then reads the variables defined in */etc/rc.conf* and initializes the services and daemons specified.

Most variable definitions in *rc.conf(5)* reflect the configuration settings entered in the Mach^{Ten} Control Panel. When you modify the Control Panel, the changes are applied to *rc.conf* the next time Mach^{Ten} boots. For definitions not covered by the Control Panel, edit *rc.conf* and modify the variable directly. All *rc.conf* definitions are described below.

The first set of variable definitions control the configuration of the network interface(s) installed in your system:

HOSTNAME	The name of the machine running Mach^{Ten}.
ADDRINFO_*	The host name of the machine on Ethernet (e), AppleTalk (a), SLIP (s), and / or TokenRing (t). All assigned names should also appear in the */etc/hosts* file.
IFCONFIG_*	The network interface configuration parameters for Ethernet (ie0), AppleTalk (at0), SLIP (sl0), and / or TokenRing (tr0). This variable is only defined if the interface is installed in the system and is to be initialized by Mach^{Ten}.
NETMASK_*	The Internet subnetwork mask for a configured network interface on Ethernet (e), AppleTalk (a), SLIP (s), and / or TokenRing (t).

The following boolean variables determine whether or not an optional service is invoked, or is started by *rc*. A null value (or string) is interpreted as negative; any other value is interpreted as affirmative, for example, the entry:

START_sendmail="yes" will start the *sendmail* daemon during the boot process, whereas the entry *START_sendmail=* will not start *sendmail*. Unused variables can be also be disabled (made negative) by preceding the entry with a pound ('#') sign.

CLEANUP_tmp	Remove files from */tmp* (aka */var/tmp*) at startup.
MOUNT_REMOTE_FILES	Mount all file systems recorded in */etc/fstab* (*fstab(5)*) at startup.
PRESERVE_EDITOR_FILES	Preserve *ex* and *vi* edits in progress in the event of a system crash.
START_cron	Execute commands by date / time via *cron(8)*.

START_inetd	Service incoming internet connections via *inetd(8*).
START_lpd	Manage spooled printer output via *lpd(8).*
START_macmntd	Automatically mount inserted floppies via *macmntd(8).*
START_named	Answer network queries for host names and addresses via *named(8)*, the domain name server.
START_sendmail	Enable incoming internet mail via *sendmail(8).*
START_syslogd	Log system messages via *syslogd(8).* By default, this variable is set if the *START_sendmail* definition is affirmative.

The following definitions are miscellaneous service definitions:

GATEWAY	The host name or internet address of the gateway to other networks. The name given should also appear in the */etc/hosts* file. The *route(8)* command is invoked with this variable by *rc* to establish the default route. The variable should remain undefined if no gateway exists.
NAMED_BOOT	The Domain Name Server boot file. By default, the file is */etc/nsdata/boo*t.

When editing */etc/rc.conf*, be careful not to remove any variables which the script */MachTen/misc/rc.config* expects to find.

As a final step in the startup process, */etc/rc* executes commands in a user-supplied file named */etc/rc.local*, if it exists. The */etc/rc.local* file is typically used to start up any machine specific processes after Mach^{Ten} has booted, such as MacTCP applications. Online examples of these files can be found in the */usr/share/skel* directory.

5.2.1 Setting Up Mach^{Ten} to Boot Automatically

Mach^{Ten} can be set up so that it boots automatically when you start your machine. Place an alias of the *MachTen* application file in the *System Folder:Startup Items* folder. Later, if you want to reset your Macintosh not to boot Mach^{Ten} automatically, simply remove the alias files from the *Startup Items* folder.

5.2.2 Manual Startup

If Mach^{Ten} has not been configured to start automatically, simply double-click the *MachTen* application in the *MachTen* folder from your Finder desktop.

5.3 Tailoring Mach^{Ten} Windows

Mach^{Ten} windows are started up by the *wind(8)* daemon which is found in the */usr/bin* directory. Note how */usr/bin/wind* is invoked by the *.login* file when you log on to the Mach^{Ten} console. When */usr/bin/wind* is initiated, the system looks in your home directory for a file named *.windrc*, which specifies your preferred window environment. If there is no *.windrc* file in your home directory, a single 24-row by 80-column window is opened with a default font and font size (Monaco 9 or the closest available fixed-width font).

If you want to design your own window environment, create a *.windrc* file in your home directory. The easiest way to create a *.windrc* file is to create all the windows you want for your standard startup environment, position them as desired, and set the active window's font type and point size to the font and size you prefer. Then select Save Window Environment in the File menu. You will be asked to provide a file name. The first time you do this, you must name the file *.windrc* file and place it in your home directory. The changes will take effect the next time you launch Mach^{Ten}. It is a good idea to keep a copy of your current *.windrc* each time you change your window environment so that you can easily return to the previous environment. If you want to return to the system default, simply remove the *.windrc* file you created.

To change the name of a terminal window, edit the *.windrc* file and change the name from "Terminal n" to whatever you would like. If you want a terminal window to run an alternative program instead of the shell specified in your */etc/passwd* entry, simply follow the window name with a *-command*, where *command* is any legal program name (you must use the complete pathname) and any parameters. When that terminal window opens it will run the command specified. Note that the terminal window will close automatically when that command terminates. Useful commands to use here are interactive programs such as */usr/bin/rlogin <hostname>* or */usr/bin/vi <filename>*. See the *wind(1)* man page for details on tailoring your *.windrc* environment.

5.4 Tailoring Your UNIX Environment

The user's environment includes the TERM, TERMCAP and TTY environment variables set by the *wind* daemon. It is important that these remain set to proper values; otherwise undesirable functioning of the VT100 emulation within your terminal windows may result. Under Mach^{Ten}, *login(1)* creates a special variable — MACHTEN, indicating the Mach^{Ten} kernel version number.

To see your current environment, type:

```
printenv
```

A typical environment looks like this:

```
VIRTUAL_MEMORY=false
POSIX=true
HOME=/home/root
SHELL=/bin/csh
LOGNAME=root
USER=root
PATH=/usr/bin:/bin:/usr/sbin:/sbin:/usr/X11/bin:/usr/libexec
MACHTEN=4
PWD=/var/tmp
EDITOR=/usr/bin/vi
VISUAL=/usr/bin/vi
EXINIT=set shell=/bin/csh magic redraw
TERM=vt100
TERMCAP=vt100:do=^J:co#80:li#24:cl=50\E[;H\E[2J:sf=2*\ED:le=^H:bs:am:cm=5\E[%i%d
;%dH:nd=2\E[C:up=2\E[A:ce=3\E[K:cd=50\E[J:so=2\E[7m:se=2\E[m:us=2\E[4m:ue=2\E[m:
md=2\E[1m:mr=2\E[7m:mb=2\E[5m:me=2\E[m:is=\E[1;24r\E[24;1H:if=/usr/share/tabset/
vt100:rs=\E>\E[?3l\E[?4l\E[?5l\E[?7h\E[?8h:ks=\E[?1h\E=:ke=\E[?1l\E>:ku=\EOA:kd=
\EOB:kr=\EOC:kl=\EOD:kb=^H:ho=\E[H:k1=\EOP:k2=\EOQ:k3=\EOR:k4=\EOS:pt:sr=2*\EM:v
t#3:xn:sc=\E7:rc=\E8:cs=\E[%i%d;%dr:
TTY=/dev/wina0
WIND_NAME=Terminal 1
DISPLAY=:0
```

To change the default editor from *vi(1)* to *ex(1)*, for example under *csh(1)*, type:

```
setenv EDITOR /usr/bin/ex
```

The environment default settings have been carefully chosen to minimize difficulties for users new to UNIX. Be careful not to unwittingly change these parameters. To return to default settings, simply restore the original *.profile* or *.login* and *.cshrc* files.

5.5 Quitting Mach$^{\text{Ten}}$

Selecting Quit from the File Menu can be used to quit Mach$^{\text{Ten}}$ without restarting the MacOS. Mach$^{\text{Ten}}$ can subsequently be launched again. The command *reboot* will kill all of the UNIX processes, and then quit Mach$^{\text{Ten}}$. The *-q* option to *reboot* will quit Mach$^{\text{Ten}}$ immediately without attempting to kill the other UNIX processes. To provide a degree of remote control (via Mach$^{\text{Ten}}$) to restart the MacOS, use *reboot* with *-M*. *Halt* behaves like *reboot*, except that after killing the UNIX processes (if no *-q* option), the Macintosh is shut off (like Finder selecting Shut Down).

When the Finder is used and either Restart or Shut Down is selected from the Special menu, Mach$^{\text{Ten}}$ invokes */sbin/reboot* to kill the UNIX processes and then quit Mach$^{\text{Ten}}$. Control returns to the Finder to Shut Down or Restart as appropriate.

When Mach$^{\text{Ten}}$ menus are used and the item Quit under the File menu is selected, Mach$^{\text{Ten}}$ invokes */sbin/reboot*. Holding the Shift key while selecting this item invokes */sbin/reboot -q*. This also holds for the hot key Command-Q and Shift-Command-Q combinations.

5.6 Login Accounts

Since Mach^{Ten} supports parallel execution of programs, it is possible to support multiple users at the same time. Each user, whether accessing the system locally or over a network, has a user name and a password. The password is used to convince the system that the users are really who they say they are. While login names are used for a variety of public purposes, such as receiving electronic mail, passwords are meant to be kept as private as possible and therefore are not echoed by the system. Having individual login accounts enables each user to have a customized working environment.

5.6.1 The Concept of a Home Directory and a User Environment

When users are given accounts on a UNIX system they are assigned individual and group IDs and are given a node in the directory tree structure. This node (also called the user's "login directory" or "home directory") designates a private storage area for that user. In addition, a "user environment" is established for each user. The user environment is governed by the user's choice of UNIX shell (see section "5.6.2 The UNIX Shell") and associated login file(s) placed in the user's home directory. The parameters that describe each user account are kept in a special password file, */etc/passwd* (see*passwd(5)*), described in the sections "5.6.7 The Password File" and "5.6.8 The Group File".

Mach^{Ten} supplies a utility program (*adduser(1)*) for establishing login accounts and home directories. (See section "5.6.6 Setting Up User Accounts".)

5.6.2 The UNIX Shell

The UNIX *shell* is the UNIX command interpreter. When you type commands in a Mach^{Ten} terminal window, you are communicating with the shell. Mach^{Ten} provides both the Bourne shell (called *sh*) and the C shell (called *csh*). If you have already logged in as *root*, you are running under the C shell (called *csh*). Both the Bourne shell and the C shell are powerful execution environments. In addition, both shells support programming languages that include control-flow primitives, parameter passing, variables, and string substitution. The Bourne is more compact and uses less memory. The Bourne shell relies on a *.profile* file in the user's home directory, while the C shell uses *.login* and *.cshrc* files to determine the user's environment. As part of establishing a user account you may stipulate a particular shell by specifying the appropriate parameter to the *adduser* script, or you may change shells at a later time using the *chpass <username>* command.

5.6.3 Superuser and Privileges

Intrinsic to each *login* is the concept of login privileges. When you are logged in as *root* you are the most privileged user on the system — the *superuser*. The superuser has access to and authority over system resources that are not available to all users. The superuser has access to private user data, can add and delete users from the system, and can provide support when a user has forgotten his or her password.

You can become superuser by logging in as root or by typing “su” when logged in as a normal user. In order for an ordinary user to become superuser, access must be granted via the */etc/group* “wheel” group.

If an ordinary user is also system administrator, it is advisable to log in as superuser only when doing administrative tasks and to revert to the normal user login otherwise. This is a security precaution to ensure that key system resources are not inadvertently destroyed. When logged in as an ordinary user, typing *su* will enable you to become superuser for performing tasks that require extra privilege.

```
$ su
Password:  your_root_password
. . . privileged operations . . .
# ^D
$
```

5.6.4 A Word About Security

Since Mach^{Ten} is implemented as an application on a non-secure operating system (MacOS), the traditional UNIX-style security is not enforced by Macintosh applications. However, since Mach^{Ten} imposes traditional UNIX permissions on files and supports traditional UNIX password protections, Mach^{Ten} files are protected from users logging in remotely and Mach^{Ten} permissions are enforced on NFS exported files. Since the Mach^{Ten} file system is simply the native Macintosh file system, traditional Macintosh security software, such as FolderBolt, works well with Mach^{Ten} for file protection. In addition, security-relevant database files, such as the password file, can be stored on remote secure UNIX servers and accessed via NFS.

5.6.5 The Root Password

In a newly installed Mach^{Ten} system, the *root* is not associated with a password. Once you log in, you should immediately invoke the *passwd(1)* program and create a password for the *root* login. This is done by typing the following:

```
#passwd
New password: <new_root_password>
Retype new password: <new_root_password>
```

As a precaution, in order to ensure that system resources will not inadvertently be destroyed, you should normally log in as an ordinary user and use the *su* (*superuser*) command to perform tasks that require extra privilege. The *su* command will give you the same capabilities as the *root* login and will expect the *root* password. Only users listed as belonging to the "wheel" group in the */etc/group* file can use the *su* command (see Figure 19).

```
%su
password: <root_password>
#                                (... whatever you need to do)
#^d                              (a control-d ends the session)
%
```

Note that the system prompt for an ordinary user is different than the *superuser* or *root* prompt. This difference will remind you to exit from *su* once administrative tasks have been completed. Read the next few sections and then set up an account for yourself and for any others that will be using the system. When you are setting up login accounts, you can give certain users the privilege of becoming *superuser*.

5.6.6 Setting Up User Accounts

A user account is added to the system using the *adduser(1)* command:

```
adduser username [-g gid] [-u uid] [-p passwd]
[-n user] [-d home] [-s login]
```

-g (gid)	Group id of the new user.
-u (uid)	User id of new user.
-p (passwd)	Text form of the user password; default is *username*.
-n (user)	Specifies user information, e.g. "John Brown"; default is *username*.
-d (home)	Specifies the user's home directory; default is */home/username*.
-s (login)	Specifies program to run as login shell; default is */usr/bin/sh* (Bourne shell). The entry */bin/csh* specifies the C shell.

Only the *username* is required; *adduser* will assign default values to the remaining parameters. A group id of 10 is suggested for user accounts. As an example, to establish an account for user Laura in group 10, enter:

```
adduser Laura -g 10
```

Adduser creates default *.login* and *.cshrc* files for the new user. A portion of the *.cshrc* file is shown in Figure 17.

Once the system recognizes a user, it sets the current working directory, executes a series of commands to perform any automatic startup processing, checks to see if any electronic mail has arrived, and prints a prompt ($)[†] indicating that the system is waiting for input.

***.cshrc* File:**

```
set    host=`hostname` tty=`tty`     Set up some additional useful shell variables

set mail=/usr/spool/mail/$USER       Tell the shell where to look for new mail.  It will
                                     check every 10 minutes and notify you if there
                                     is anything new.
if ($?prompt) then                   Run only the following on interactive shells.
    cd                               Change directory to home directory every time.
    setenv EDITOR '/usr/ucb/vi'      Set Mail control variables.
    setenv SHELL /bin/csh
    setenv VISUAL /usr/ucb/vi
    setenv EXINIT 'set shell=/bin/csh magic redraw'setenv SHELL /bin/csh
                                     ex/vi settings

                      SET UP CSH CONTROL VARIABLES
    set filec                        Enable filename completion with <esc> and ^d.
                                     Normally csh will log out when ^d is typed at
                                     start of command line.
    set ignoreeof                    Disable this feature.
    set noclobber                    Tell csh not to write over existing files during
                                     I/O redirection.
    set notify                       Tell csh to provide immediate job completion
                                     messages.
    set history=20                   Tell csh to remember the last 20 commands.
    set savehist=20                  Tell csh to remember last 20 commands from
                                     previous login.
    set prompt="$user@$host% "       Set prompt to include the user's name and local
                                     host's name.

                        COMMAND ALIAS SECTION
    alias psh pushd                  Set up some useful short versions  of often
    alias pop popd                   used commands.
    alias hi history
    alias j jobs -l
    alias cd 'set old="$cwd"; chdir \!*'
    alias back 'set back="$old"; set old="$cwd"; cd "$back"; unset back; dirs'
                                     Set up alias for cd/back.  cd works like regular
                                     cd and sets shell variable $old so that the back
                                     alias will put us back to the previous directory.
    alias new 'ls -aglt \!* | head -15'
                                     Set up alias that will show the 15 newest files in
                                     the local directory, newest first.
endif
```

Figure 17. user .cshrc File

[†] The '$' prompt is the default prompt for *sh(1)* users; for *csh(1)* users, the default prompt string includes your user name, host name and the '%' character in the form: *username@host%*

5.6.7 The Password File

When set up as a user, you should add a password using *passwd.*

In 4.4BSD, the */etc/passwd(5)* file is an ASCII file which is a shadow of the actual password database file. Do not make changes to this file. Use the *passwd, chpass,* and *chroot* commands to make changes to the actual password database, which will automatically be reflected in the */etc/passwd* file!

The */etc/passwd(5)* ASCII file contains entries for the following:

name (login name, contains no uppercase)
encrypted password
numerical user ID
numerical group ID user's real name, office extension, home phone
home directory
program to use as shell

Each field is separated by a colon. Each "user" is separated by a new-line. If the password field is null, then no password is demanded. If the password field consists of a single asterisk, the user account is considered not valid for login. This convention is used for several system level accounts and for old user accounts that are in the process of removal. If the shell field is null, then */bin/sh* is used. If a user forgets his or her password, the system administrator can change their password using the *chpass* command.

Each user should have a unique user ID. In a networked environment, it is often most useful to match user IDs between systems so that users can easily access files via the Network File System (NFS). Look at the */etc/passwd* file on the other systems to determine what the user ID should be for each user. The user ID that you use to log in to the system console is the one used for permission mapping during Desktop NFS access to other file systems, even if you subsequently use *su* or *login* to change to some other user.

Notice also that *root* has selected the C shell. By using the *chpass* command, you can change the *root* default shell to the Bourne shell by using the *chpass* command.

```
root:*:0:1:Operator:/home/root:/bin/csh
toor:*:0:1:Bourne-again Operator:/home/root:/bin/sh
nobody:*:65534:65534:Unprivileged user:/tmp:/sbin/nologin
daemon:*:1:1::/:/sbin/nologin
sync:*:1:1::/:/bin/sync
operator:*:2:5:System &:/home/operator:/bin/csh
bin:*:3:7:Binaries Commands and Source:/usr/bin:/sbin/nologin
uucp:*:4:4:UNIX-to-UNIX Copy:/var/spool/uucppublic:/usr/lib/uucp/uucico
news:*:6:8:Network News:/var/spool/news:/sbin/nologin
games:*:7:13:Games pseudo-user:/var/spool/news:/sbin/nologin
ftp:*:19:19:Anonymous FTP:/home/ftp:/sbin/nologin
```

Figure 18. The /etc/password File

5.6.8 The Group File

The */etc/group* file is used to organize users into groups in order to control system access and authorizations. Groups are defined by entries in the */etc/group* file. Each entry has the following fields:

name
encrypted password
numeric group id
authorized users (separated by commas)

The fields are separated by colons. The authorized user list is optional. It provides a way for users to belong to multiple groups. The encrypted password is usually an asterisk, preventing use by means of a password.

Figure 19 shows how the */etc/group* file looks when Mach^{Ten} is first installed. This is the file used to designate group affiliations. The first line in the */etc/group* file is the "wheel" entry, for people with total access to system resources or *root* privileges — the "big wheels". Only users listed for "wheel" group are allowed to become superuser with the *su(8)* command. You may want to establish separate logins for members of a project team and give all members access to the project files. An entry for an accounting group with two members would be as follows:

```
accounting:*:7:shawn, lauren
```

```
wheel:*:0:root
nogroup:*:65534:
daemon:*:1:daemon
kmem:*:2:root
sys:*:3:root
tty:*:4:root
operator:*:5:root
bin:*:7:
news:*:8:
wsrc:*:9:
games:*:13:
dialer:*:17:
ftp:*:19:
staff:*:20:root
```

Figure 19. The /etc/group File

5.6.9 Administrative Login Accounts

It is customary to set up special administrative accounts with varying degrees of privilege. These accounts are used by system administrators or by programs to perform specific administrative tasks. The default Mach^{Ten} */etc/passwd* file contains the following accounts:

```
root:*:0:1:Operator,,,:/home/root:/bin/csh
toor:*:0:1:Bourne-again Operator:/home/root:/bin/sh
nobody:*:65534:65534:Unprivileged user:/tmp:/sbin/nologin
daemon:*:1:1::/:/sbin/nologin
sync:*:1:1::/:/bin/sync
operator:*:2:5:System &:/home/operator:/bin/csh
bin:*:3:7:Binaries Commands and Source:/usr/bin:/sbin/nologin
uucp:*:4:4:UNIX-to-UNIX Copy:/var/spool/uucppublic:/usr/lib/uucp/uucico
news:*:6:8:Network News:/var/spool/news:/sbin/nologin
games:*:7:13:Games pseudo-user:/var/spool/news:/sbin/nologin
ftp:*:19:19:Anonymous FTP:/home/ftp:/sbin/nologin
```

Non-interactive accounts for programs:	
nobody	Default user id for remote root access via NFS.
daemon	Owner of certain background processes that handle system services, such as print spooler (*lpr(1)*) and network communication.
sync	Special login to update write buffers. Performs a *sync* and logs out.
bin	Owner of most standard commands.
uucp	Login assigned to *uucp* requests.
news	Login used by *netnews* program. This is a placeholder since *news* is not implemented in Mach^{Ten}.
User accounts:	
root	Login for superuser. The home directory is / and the shell is */bin/csh.*
toor	Alternate login for superuser. The home directory is / and the shell is */bin/sh.*
Sjoe	Example entry for a remote SLIP account.

5.6.10 Writing to All Users

Wall(1) reads its standard input until an end-of-file. It then sends the message it read, preceded by “`Broadcast Message ...`”, to all users. The sender should be superuser to override any protections that users may have invoked.

```
#wall
The system is going down in 5 minutes for maintenance.
^D
```

The result on each user’s active terminal window will be:

```
Broadcast message from root:
The system is going down in 5 minutes for maintenance.
```

5.6.11 Disabling User Logins

If the file */etc/nologin* exists, no logins (other than *root*) will be allowed on your system. See *login(1)*.

5.6.12 Removing Users

Removing a user involves removing the entry from the */etc/master.passwd* file, rebuilding the password databases with *pwd-mkdb* and deleting disk files. This may be a phased process. To prevent a user from logging in, use the *chpass* command and change the password string for the user, thus preventing any possible match. Then, when convenient, perhaps after backing up, the deleted user’s storage can be reclaimed.

5.6.13 Changing the Message-of-the-Day

As *root* or *su* you can edit the file */etc/motd*. If this file does not exist it can be created with the *touch(1)* command. Whatever data you place in that file will be printed out each time a user logs into Mach^{Ten} (except for the top two lines, which are edited by the */etc/rc* script at reboot and filled in with a version number).

5.6.14 Special Characters

A number of keyboard characters have special meaning in UNIX. The Special Keyboard Characters chart in Figure 20 will help you use the UNIX command line environment.

TERMINAL HANDLER SPECIAL CHARACTERS

• Line Termination	The Return key indicates completion of input at the end of a line. A command is initiated when the Return key is struck.
• Character Deletion	The Delete key deletes a character.
• Word Deletion	Control-W deletes the last word of input.
• Line Deletion	Control-U deletes the entire line.
• Output Suspend / Restart	Control-S temporarily halts output to the screen. Control-Q restarts output.
• Program Interrupt	Control-C interrupts a program. Programs that do not intercept this input are terminated and the command interpreter is given control.
• End of File	Control-D demarks the end of a series of input lines.
• End of Session	Control-D is used by one of the command interpreters to demark end of session and log a user out of the system. (sh only)
• Program Suspension	Control-Z suspends execution of interactive programs and takes a user to the command interpreter. fg (foreground) returns the user to the program. (csh only)
• Escaping Input	Preceding special characters with a "\" (backslash) escapes the special character and passes it on as regular input.

COMMAND INTERPRETER SPECIAL CHARACTERS

• *	Matches zero or more characters in a filename. May be used to match filename strings.
• ?	Matches one single character pattern.

Figure 20. Special Keyboard Characters

5.7 Managing Your UNIX Environment

5.7.1 What's Running?

Since Mach^{Ten} runs multiple programs at the same time, there will be times when you will want to see which programs are executing, how much system time they are using, and, if they are not running, why not. The program *ps* prints the process status of each of the Macintosh and UNIX programs that are running. Figure 21 shows the sample output from a *ps* command.

5.7.2 Killing a Program

The *kill(1)* program is used in conjunction with the *ps* program to abort the execution of a program. First, use *ps* to determine the process id of the program that you wish to abort. Then use the *kill* program with that process id to terminate that program. Occasionally, after you execute *kill*, you may receive a message telling you that the program actually terminated. This delay results from the fact that to terminate a program, the program must be scheduled for execution and must terminate itself. For example,

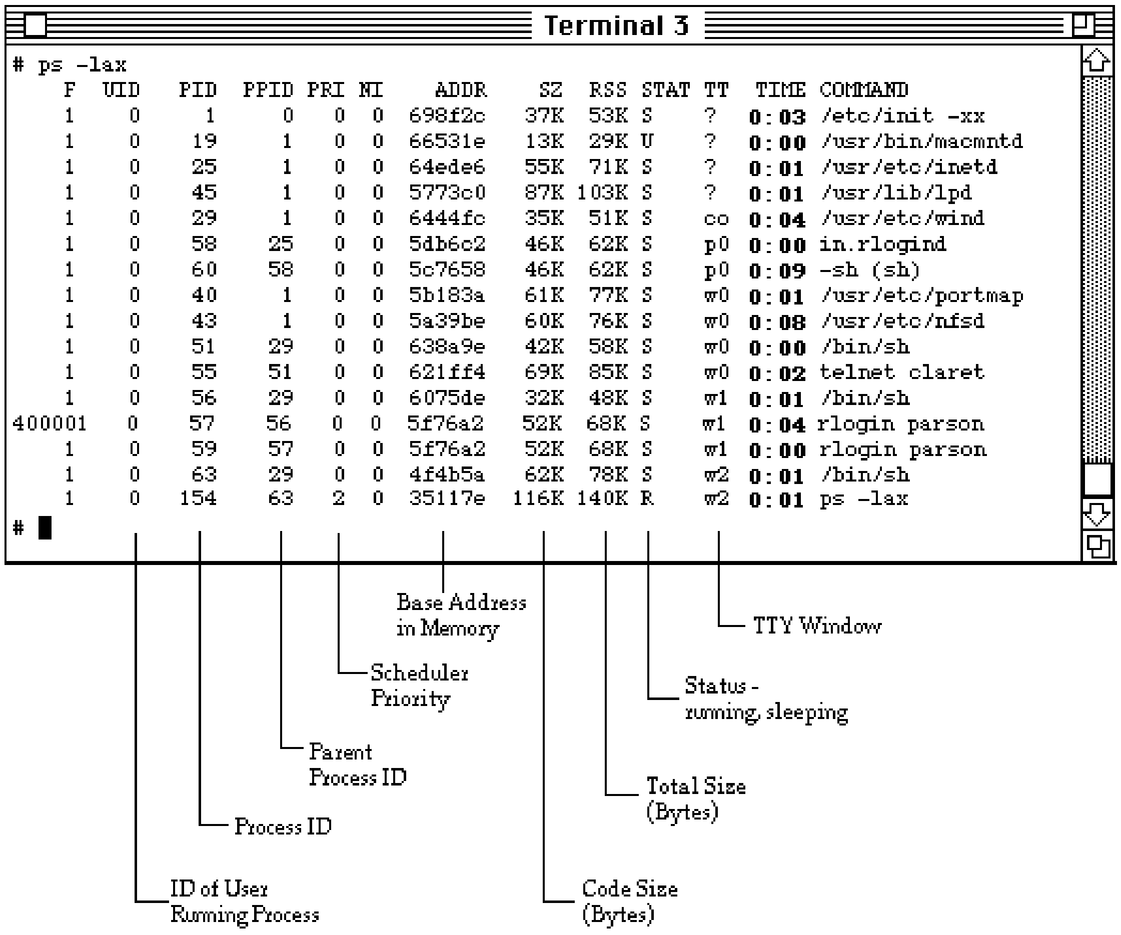
```
kill -9 25
```

would abort the *inet* daemon shown running in Figure 21.

5.7.3 Background Program Execution

Some programs execute for a long time. Rather than tie up a window, you can start the program and let it run behind the scenes. To place a program in "background execution", simply type an ampersand (&) at the end of the command line. The shell will begin executing the program and, rather than waiting for the program to complete execution before returning a prompt, the shell returns immediately with a process id identifying the program that started execution, and a prompt signifying that it is ready to run another command. For example:

```
% grep ioctl *.c > found &
```



```
# ps -lax
     F  UID    PID   PPID  PRI  NI     ADDR     SZ   RSS  STAT  TT   TIME  COMMAND
     1    0      1      0    0   0  698f2c     37K   53K  S     ?    0:03  /etc/init -xx
     1    0     19      1    0   0  66531e     13K   29K  U     ?    0:00  /usr/bin/macmntd
     1    0     25      1    0   0  64ede6     55K   71K  S     ?    0:01  /usr/etc/inetd
     1    0     45      1    0   0  5773c0     87K  103K  S     ?    0:01  /usr/lib/lpd
     1    0     29      1    0   0  6444fc     35K   51K  S     co   0:04  /usr/etc/wind
     1    0     58     25    0   0  5db6c2     46K   62K  S     p0   0:00  in.rlogind
     1    0     60     58    0   0  5c7658     46K   62K  S     p0   0:09  -sh (sh)
     1    0     40      1    0   0  5b183a     61K   77K  S     w0   0:01  /usr/etc/portmap
     1    0     43      1    0   0  5a39be     60K   76K  S     w0   0:08  /usr/etc/nfsd
     1    0     51     29    0   0  638a9e     42K   58K  S     w0   0:00  /bin/sh
     1    0     55     51    0   0  621ff4     69K   85K  S     w0   0:02  telnet claret
     1    0     56     29    0   0  6075de     32K   48K  S     w1   0:01  /bin/sh
400001    0     57     56    0   0  5f76a2     52K   68K  S     w1   0:04  rlogin parson
     1    0     59     57    0   0  5f76a2     52K   68K  S     w1   0:00  rlogin parson
     1    0     63     29    0   0  4f4b5a     62K   78K  S     w2   0:01  /bin/sh
     1    0    154     63    2   0  35117e    116K  140K  R     w2   0:01  ps -lax
#
```

Figure 21. Using ps to Manage Your Environment

5.7.4 Shell Files

Many times you will find that you are executing a series of commands in the same order with roughly the same parameters. Rather than typing that set of commands each time, it would be useful to combine the commands into a single file and hand the set of commands to the shell for execution by simply specifying the file name. UNIX shell files provide that capability. Use the editor to create a file called *startup* with the commands *date, who,* and *pwd*. Here is an example using the *ed* editor:

```
%ed                 (invoke ed)
#!/bin/sh
a                   (append text)
date
who
pwd
w startup           (write to a file named startup)
q                   (quit)
%
```

To execute this file type:

```
%sh startup
```

Note that the commands are executed before the prompt is returned. If you change the file permissions to allow the file to be executed, you can eliminate the "sh". Type:

```
%chmod +x startup
```

This changes the mode of the *startup* file and makes it an executable file.

Now you can simply type:

```
%startup
```

and the commands within *startup* will execute. If you place *startup* in your *.login* or *.profile* file (depending upon your standard shell), these commands will be executed every time you log in.

The shell has a sophisticated command language to allow you to control and parameterize the execution of a set of programs.

6.0 The Mach^{Ten} File System

In the traditional UNIX world, disks are *devices* that can be formatted into file systems and *mounted* for access by UNIX applications. In the Macintosh world, disks are called volumes. The Macintosh File Manager is the entity responsible for formatting disks into volumes and controlling access to the files on those volumes.

Mach^{Ten} implements the UNIX file system on top of the Macintosh file system. UNIX files are stored on Macintosh volumes alongside the Macintosh files. An advantage of this design is that there is no need to reformat your disk into a UNIX partition and a Macintosh partition.

Mach^{Ten} imposes UNIX file system semantics on the Macintosh by mapping the UNIX file system requests onto the appropriate Macintosh File Manager routines. The major differences between the Macintosh file system and the UNIX file system are noted under “Differentiating UNIX and Macintosh Files”.

6.1 File System Structure

Mach^{Ten} conforms to the Berkeley file system (sometimes called UFS). The Mach^{Ten} file system is organized as a tree-structured hierarchy of directories, device nodes, and ordinary files.

At the top of the Mach^{Ten} File System hierarchy is the UNIX *root* directory, also referred to as “/”. When you double-click on your hard disk and view your folders and files using Finder, you are viewing the *root* directory. A UNIX directory is simply a Macintosh folder.

6.1.1 The root Directory Tree

The *root* directory is represented by a / and contains the following directories:

CDROM	mount point for Mach^{Ten} CD-ROM
MachTen	Mach^{Ten} application and miscellaneous configuration files
base	base of source code and place for building binaries in Mach^{Ten} distribution
bin	basic user utilities
dev	block, character and other special device files
etc	system configuration files and scripts
home	contains user home directories
mnt	temporary mount point
sbin	system binary files
tmp	directory for temporary files
usr	contains majority of system utilities and files
var	multi-purpose log, temporary, transient and spool files
volume	*macmntd* creates subdirectory mount points for removable volumes

6.1.2 Floating Root

Power Mach^{Ten} supports a UNIX File System root that is not necessarily at the corresponding root of the Macintosh File System. The Mach^{Ten} file system root may be in any folder on a Macintosh HFS disk. This capability has been dubbed "Floating Root". Note that the Floating Root should not be moved or changed while Mach^{Ten} is running. Also, the relative positions of the files and folders within the Mach^{Ten} Root Folder should not be changed.

When a UNIX process accesses the path "/", it will refer to the Macintosh Folder which is at the root of the Mach^{Ten} hierarchy (typically named *MachTen Root*). The path "//" is a special case in Mach^{Ten} that allows UNIX processes to refer to files and folders outside of the Mach^{Ten} root. "//" is interpreted to mean "the real root of the HFS volume that contains the Mach^{Ten} file system hierarchy". The corresponding routines in the Mach^{Ten} libraries (*getwd(3)*) have been modified to understand this "//" special case.

6.1.3 The usr Directory Tree

X11	symbolic link to Mach^{Ten} X Window software
bin	binaries
include	*include* files
info	online information for some of the GNU applications
lib	libraries
libexec	catchall for system daemons
local	directory for applications created locally
macppc	PowerPC specific files go here
sbin	system administrator binary files
share	text and data base files readily shared among 4.4BSD systems

6.1.4 The var Directory Tree

adm	administrative files
at	timed command scheduling files
backups	miscellaneous backup files
cron	*cron* data files
db	database files
games	miscellaneous game status and log files
log	miscellaneous system log files
mail	user system mailboxes
msgs	system messages
obj	object files
preserve	temporary home of files preserved when editors fail
run	system information files, rebuilt after each reboot
rwho	*rwho* data files
spool	miscellaneous printer and mail system spooling directories
tmp	temporary files that are not discarded between system reboots

6.2 Major System Administration Files

For details about these files, see online man pages.

/etc/aliases	defines mailbox aliasas and mailing lists for *sendmail(8)* [*aliases(5)*]
/etc/exports	specifies what file systems (directory hierarchies) are exported to NFS clients [*exports(5)*]
/etc/fstab	contains file system mount information
/etc/group	defines mapping of group numbers to names and authorized users [*group(5)*]
/etc/hosts	defines host names, aliases and IP address
/etc/hosts.equiv	contains the list of host names from which users may remotely log in without passwords (needs to be created if you want to support *rlogin* or printer spooling from remote machines)
/etc/hosts.lpd	needs to be created if you want to support printer spooling from remote machines without allowing *rlogin* access
/etc/inetd.conf	specifies services *inetd* should support [*inetd(8)*]
/etc/networks	defines network names and numbers; may be symbolic link to */usr/etc/networks* [*networks(5)*]. Large networks like the Internet now use domain name servers, and the */etc/networks* file typically is not used.
/etc/net_search_rules	contains database search rules for resolving host names to IP addresses [*net_search_rules(5)*]
/etc/passwd	defines the set of users authorized to log in; provides mapping between user id and name, default group, home directory, and shell [*passwd(5)*]
/etc/printcap	contains the list of available printers and the parameters used by the line printer daemon [*printcap(5)*]
/etc/rc	contains the script of commands *init(8)* is to execute to establish the desired environment [*rc(8)*]
/etc/rc.conf	contains variables derived from information entered in the Mach^{Ten} Control Panel for use by */etc/rc* when starting Mach^{Ten}
/etc/remote	describes remote systems involved in serial line connections, including SLIP connections [*remote(5)*]
/etc/resolv.conf	contains domain name information read by Mach^{Ten} domain name resolver routines [*resolver(5)*]

/etc/sendmail.cf	*/usr/lib/sendmail* configuration file [*sendmail(8)*]
/etc/syslog.conf	*/usr/etc/syslogd* configuration file [*syslog(8)*]
/etc/termcap	defines the capabilities for numerous terminal types [*termcap(5)*]
/etc/ttys	specifies terminal support information for *init(8)* [*ttys(5)*]

6.3 File Names

Mach^{Ten}’s implementation of the local UNIX file system (UFS) uses the Macintosh File Manager (HFS), enabling Mach^{Ten}’s UNIX files to be stored alongside Macintosh files. However, the rules for file and directory names permitted by the Macintosh File Manager differ from the rules for names in traditional UNIX systems.

6.3.1 Maximum Number of Characters

The Macintosh File Manager allows a maximum of 31 characters in a file name; hence, Mach^{Ten} file names are limited to 31 characters.

6.3.2 Case Sensitive File Names

Mach^{Ten} supports case sensitive file names on the case insensitive Macintosh File System (HFS) by appending a 4 byte integer to the end of any case sensitive file name that collides with an existing case insensitive name. HFS stores the length of the file name, hence the names are unique within HFS. UNIX requests use a null terminated string comparison for identifying files, hence the appropriate file is matched as long as the first of the four appended bytes is NULL. Mach^{Ten} guarantees that this is the case. For subsequent collisions, a new integer is used.

Since the HFS limit on file name length is 31 characters, this method of resolving case sensitivity is limited to file names of length 27 characters or less. Attempting to create a new file that collides with an existing case insensitive file name of greater than 27 characters will fail.

6.3.3 Component Separators

The Macintosh File Manager uses the character ‘:’ to separate names into volume, folder, and file name components; hence, Macintosh file names cannot contain that character. UNIX uses the character ‘/’ for the same purpose. Mach^{Ten} automatically maps all occurrences of ‘:’ (which is valid in UNIX file names) into ‘/’ (which is valid in Macintosh file names) before passing UNIX file names to the Macintosh File Manager.

6.3.4 Non-Printable Characters

Macintosh file names often include unprintable characters (the trademark symbol, for example). Although these characters are not invalid in UNIX file names, they are impossible to type or display on a UNIX command line. For this reason, non-printable ASCII characters in Macintosh file names are translated according to the AppleSingle 7-bit ASCII naming convention.† Basically, non-printable characters are translated to a percent sign (%) followed by a 2 digit hexadecimal representation of the character's value. The following are some examples:

File Name as Viewed by MacOS	File Name as Viewed by Mach^{Ten}
Filename™	Filename%aa
Filename®	Filename%a8
Filename©	Filename%a9

In the delivered Mach^{Ten} file system, there are no UNIX file names longer than 31 characters, nor file names that conflict due to the case sensitivity limitation. These situations are unlikely to occur often in practice, but the astute Mach^{Ten} user must be aware of these file name limitations when porting other UNIX applications to Mach^{Ten}.

6.4 Linked Files

In UNIX, linking allows several file names to be associated with the same physical file. Power Mach^{Ten} properly supports UNIX soft links and hard links via *link(2).*

6.4.1 Hard Links

Mach^{Ten} provides a file system paradigm that permits copying or moving either UNIX or Macintosh files with either UNIX or Macintosh tools with equal results. However, the UNIX implementation of hard links prohibits cross disk hard links. Using the Finder to copy Mach^{Ten}'s hard links from one volume to another will lead to unexpected results. Therefore, you cannot use the Finder (or any other Macintosh tool) to copy or move Mach^{Ten} hard links from one Macintosh volume to another. This restriction is similar to the traditional UNIX restriction — no cross disk hard links.

† For more information on AppleSingle, refer to the *AppleSingle Encapsulation* section 6.11.1.

Mach^{Ten}'s 4.0 implementation of hard links uses a hidden folder at the *root* level of a volume, and uses the Macintosh HFS equivalent of *inode* numbers to find targets of hard links in this folder. When the source of a hard link is moved to another volume, the link will be unresolvable. Even if the correct target of the link is copied from the source volume of the destination volume's hidden folder, it will be assigned a new *inode* number and the source link will still be unresolvable.

In general, unless an application requires specific semantics of hard links that are not also supplied by soft links, soft links should be the preferred method of linking files under Mach^{Ten}.

6.5 Locked Files

The Macintosh File System has the ability to lock individual files, effectively making the file 'read only'. The UNIX file system has no notion of locking for individual files, although it does support the locking of entire file systems by mounting them as 'read only'. If a UNIX application attempts to modify or remove a locked file, Mach^{Ten} will return the error EROFS. Although this error implies that the entire file system is 'read only', it may be only the individual file that is locked. In this case it is necessary to unlock the file (use the Finder's Get Info entry in the File menu) before Mach^{Ten} can modify or remove the file.

6.6 Directory Link Counts

The *stat()* system call does not return the correct number of links for a directory. The Macintosh File Manager does not keep a directory link count for Macintosh folders, and Mach^{Ten} does not attempt to fabricate a correct directory link count for UNIX directories on local Macintosh File Systems[†]. (Of course, the link count is correct for Network File System (NFS) directories.)

6.7 File Types

The Macintosh File Manager saves a file type and file creator as part of the attributes for every file. Mach^{Ten} uses the identifier 'MUMM' for UNIX files. In Mach^{Ten} documentation the term "UNIX file" is used to refer to any file with a creator 'MUMM' and the term "Macintosh file" is used to refer to all other files. Note that Macintosh applications can see and access the UNIX files, since they are simply Macintosh files with a creator 'MUMM'. Only Mach^{Ten} makes the distinction between UNIX files and Macintosh files.

All Mach^{Ten} documents and files are owned by the application named *MachTen*. Mach^{Ten} uses file types for the different classes of files in UNIX. Each file type has a unique icon, easing identification of UNIX files when using the Macintosh Finder.

[†] Mach^{Ten} always returns a link count of 2 for directories.

BINA — Binary Files

Binary files are streams of data. They may be UNIX executable files, text files, database files, or others. No translation of the data is done on binary files when accessed via Mach^{Ten}. This is the simplest, fastest, and most common type of file. When Mach^{Ten} creates a new file, it always creates files of type BINA.

TEXT — Text Files

Files containing only ASCII printable characters are referred to as "text" files. The Macintosh and UNIX formats for text files have subtle differences. Macintosh uses the character 'cr' (0xD) to terminate a line of text and UNIX uses the character 'nl' (0xA). Mach^{Ten} files with a type 'TEXT' are stored on the disk in the native Macintosh text file format (the line terminator is 'cr').

When UNIX applications access type TEXT files, Mach^{Ten} automatically translates the 'cr' character to 'nl', so the UNIX applications see these files in the expected UNIX text file format. A result of this automatic translation is that both Macintosh word processors and UNIX editors can be used to edit 'MUMM' / 'TEXT' files. Note that some Macintosh word processors may inadvertently add resource forks to the UNIX text files, but these resource forks will always be ignored by Mach^{Ten}. For more information on the handling of text files in Mach^{Ten}, refer to the chapter "Text File Manipulation".

LINK — Symbolic Links

This file type is used to identify UNIX symbolic links. The path to follow for the symbolic link is stored in the data fork of the file.

CHR — Character Devices

This file type is used to identify UNIX character devices. UNIX assigns two numbers to each device. The 'major' number describes the device type, and the 'minor' number uniquely identifies the device. The major and minor numbers for this device are packed in four bytes and are stored in the data fork of the file.

BLK — Block Devices

This file type is used to identify UNIX block devices. The major and minor numbers for this device are packed in four bytes and are stored in the data fork of the file.

SOCK — Sockets

This file type is used to identify UNIX sockets. Sockets are communication endpoints used for sending and receiving data.

FIFO — Pipes

This file type is used to identify named UNIX pipes. A pipe is a special type of file that is created by UNIX processes in order to pass information to other processes. Pipes enforce a first-in, first-out (FIFO) mechanism on data.

HLNK — Hard Links

This file type is used to identify hard links. Hard linking allows several file names to be associated with the same physical file.

SHLB — Shared Libraries

This file type is used to identify a shared library file. Many of the traditional UNIX shared libraries (*libc*, *libm*, etc.) will have two versions in Mach^{Ten} — one for the comple time Header Call definitions and the other for the run time dynamic linking.

6.8 File Permissions

Each file and directory has a set of UNIX access rights enforced by Mach^{Ten}. Access rights are divided into read, write, and execute permissions (rwx). (See Figure 22.) Read permission allows a user to examine the contents of a file. Write permission allows a user to change or append data to a file. Execute permission for binary files allows a user to execute the file as a program, and for directories allows a user to search the directory. File permissions are designated for the owner of a file, for the members of the owner's group, and for "everyone else". Mach^{Ten} stores standard UNIX user id, group id, and mode bits for each UNIX file. Mach^{Ten} can set these attributes for Macintosh files as well.[†] Although this protection is not effective when accessing either Macintosh or UNIX files locally from the Finder or other Macintosh applications, this protection is effective if the Mach^{Ten} system is exporting Macintosh or UNIX files via NFS.

Mach^{Ten} stores the protection attributes for a directory in a special file within the folder. This file has the name '/' and is invisible to the Finder. However, other Macintosh applications may see this file. It should not be deleted or changed.

```
# ls -la
total 3391
drwxr-xr-x  2 bin          9216 Apr 28 15:55 .
drwxr-xr-x  2 bin          1024 Apr 17 22:51 ..
-rwxr-xr-x  1 bin         25576 Apr 24 08:45 acp
-rwxr-xr-x  1 bin         14597 Apr 18 09:32 apply
-rwxr-xr-x  1 bin         38373 Apr 18 09:32 ar
-rwxr-xr-x  1 bin         87161 Apr 18 09:35 awk
-rwxr-xr-x  1 bin         12807 Apr 18 09:32 basename
-rwxr-xr-x  1 bin         34108 Apr 18 09:32 bc
-rwxr-xr-x  1 bin         21711 Apr 18 09:32 cal
-rwxr-xr-x  1 bin         20173 Apr 18 09:32 cat
```

Figure 22. Example of File Permissions

6.9 Time Stamps

The Macintosh File Manager saves three time stamps for each file: (1) the time the file was created, (2) the time file was last modified, and (3) the time the file was last backed up (rarely used in the Macintosh world). UNIX also requires three time stamps for each file: (1) the time the file's data was last modified, (2) the time the file's attributes were last modified, and (3) the time the file was last accessed (i.e., the data was read). When Mach^{Ten} examines the time stamps of a Macintosh file, the file's Macintosh creation time is used for the UNIX data modification time. The file's Macintosh modification time is used for the UNIX attribute modification times. The file's Macintosh modification time is set to the UNIX attribute Macintosh time. The file's Macintosh last-backed-up time is used for the UNIX last-accessed-time. When Mach^{Ten} modifies the

[†] The Mach^{Ten} application has resources which allow the user to configure the default UNIX access privilege modes for Macintosh files and folders. See the *Configuration Resources* technical note in the */pub/tech_notes* directory on *ftp.tenon.com*.

time stamps of a Macintosh file, the file's Macintosh creation time is set to the UNIX data modification time. The file's Macintosh last-backed-up time is set to the UNIX last-accessed-time.

FILE TIME STAMPS

Macintosh	UNIX
file created	<--> data last modified
file modified	<--> attributes last modified
file backed up	<--> data last accessed

Since most of the interaction between UNIX and Macintosh programs depends on only the "modified" timestamp, the Mach^{Ten} interpretation of the Macintosh timestamps provides Mac / UNIX interoperability. For example, changing a source file with a UNIX tool (like *vi*) results in a change to the file's "modified" timestamp. A Macintosh application which subsequently accesses this file will notice the change in the modified time, and will behave accordingly (e.g., a Macintosh development tool will rebuild a project if the project depends on the modified source file). In the reverse situation, a Macintosh editor will also change the file's "created" and "modified" timestamp, which will be interpreted by UNIX programs in the proper, "the-data-(or attribute)-has-been-modified" sense.

6.10 Mounting Macintosh Volumes

Mach^{Ten} stores UNIX files on Macintosh volumes alongside Macintosh files. Since all Macintosh volumes are formatted with the Macintosh Hierarchical File System (HFS), is not necessary to use the traditional UNIX file system formatting commands (*mkfs*, *fsck*, etc).

In Mach^{Ten}, the *mount(8)* command has been modified to accept a Macintosh volume name instead of the UNIX device name for mounting local file systems. When Mach^{Ten} is started, the folder from which Mach^{Ten} was launched is automatically mounted as the *root* file system ('/').

Once Mach^{Ten} is running, any other Macintosh volume may be mounted. For example, to mount an external hard disk named "External Disk" on the path */external* and to mount a floppy disk named "Untitled" on the path */volume/Untitled*, the following commands may be used:

```
mkdir  /external
mount "External Disk"   /external
mount Untitled   /volume/Untitled
```

Note that the path where the volume is to be mounted must exist (*/volume* is created by the Mach^{Ten} installation program), and that volume names with spaces must be contained within quotes.

UNIX provides a database file (*/etc/fstab*) to specify the path on which each file system is to be mounted. In Mach^{Ten}, the format of this file has also been extended to accept Macintosh volume names, just like the *mount* command. (See *fstab(5)*.)

6.10.1 Mounting Permissions

You need to be *root* or *su* in order to *mount* and *umount* floppies. This is consistent with the traditional UNIX *mount* restrictions and prevents unauthorized users (perhaps logged in over the network) from unmounting your devices. Make sure that you are not in the mounted directory while unmounting, since file systems in use cannot be unmounted. Since *macmntd* is a daemon with *root* permissions, the automatic mounting and unmounting of removable media will succeed regardless of which user is logged in on the console.

6.10.2 Automatic Mounting of Removable Media

Mach^{Ten} provides a special UNIX daemon — *macmntd(8)* — to handle automatic mounting of removable media like floppies, CD’s, and Bernoulli disks. *Macmntd* watches for disk inserted events and passes the volume name of the inserted disk to the *mount* command. If an entry for this volume exists in the */etc/fstab* file, the volume will be automatically mounted on the specified path. If no entry exists, *macmntd* will create a path named */volume/<volume name>* and mount the volume on this path.

Once mounted, whether automatically or explicitly by the *mount* command, the volume can be used as a normal UNIX file system. If you look in the */volume* directory, you will see the *floppy_name* directory. To change to the directory to see the files on your floppy, the commands are as follows:

```
cd /volume/<volume_name>
ls
```

Now you can perform any valid operation on those files, such as copying (*cp(1)*), moving (*mv(1)*), or executing (by typing the name of the file followed by a carriage return).

6.10.3 Unmounting Macintosh Volumes

In Mach^{Ten}, the *umount(8)* command has been modified to accept Macintosh Volume Names for unmounting file systems. To unmount the volumes mounted in the example above, use:

```
umount "External Disk"
umount Untitled          (floppy will not eject)
```

6.10.4 Automatic Unmounting

In Mach^{Ten}, Macintosh volumes will automatically be unmounted when the Finder is used to drag the disk to the Trash.

6.10.5 Formatting Floppies

In UNIX it is often desirable to write directly to a device (disk) without regard for file system structures. Mach^{Ten} permits this type of raw access. Under Mach^{Ten} you can format a floppy for use as a raw device. If you do not give the floppy a Macintosh volume name, the floppy will automatically be formatted as a raw device. If Mach^{Ten} is the front application when you insert an unformatted floppy, a dialog box will prompt you to supply a name; simply choose Cancel. If, instead, you name the floppy, it will get a Macintosh volume name. Beware that using a null name results in a floppy with "???" for a Macintosh volume name. Raw floppies are not seen by the Finder, so no icon will appear on the Finder desktop.

To eject a raw floppy, use the *eject(1)* command. The *eject* command is:

```
eject 1        (floppy will eject)
```

Mach^{Ten} devices for accessing floppies are */dev/rfd0* and */dev/rfd1* to provide traditional raw access, and */dev/fd0* and */dev/fd1* to buffer floppy accesses in 512 byte blocks. You can determine what number device you have by experimenting with the *eject* command. A Macintosh with a single floppy drive typically uses the *fd1/rfd1* device. Raw floppies may be accessed via *tar(1)*, for example, but they cannot be mounted.

6.11 Accessing Macintosh Files from UNIX Applications

6.11.1 AppleSingle Encapsulation

In contrast to UNIX files which have only one stream of data, Macintosh files may have two streams (or forks) of data. These streams are called the data fork and the resource fork. In order to preserve all of the data in a Macintosh file when it is accessed by a UNIX application, it is necessary to encapsulate the two streams of data into a single stream. Consider the case of using the UNIX *cp(1)* command to copy a Macintosh file. It is most desirable that both forks of the file be copied from the source to the destination. Apple has defined a standard for the encapsulation of Macintosh files into a single stream known as the AppleSingle format. Mach^{Ten} uses this method of encapsulation.

6.11.2 Differentiating UNIX and Macintosh Files

In order to determine whether or not a file must be encapsulated in the AppleSingle format, it is necessary to be able differentiate UNIX files from Macintosh files. Since Mach^{Ten} uses the file creator 'MUMM' for all UNIX files, all files with any creator other than 'MUMM' must be Macintosh files. Therefore any file with a creator other than "MUMM' is automatically encapsulated in the AppleSingle format by Mach^{Ten} when the file is accessed by UNIX applications. Conversely, when an AppleSingle format file is written by a UNIX application, Mach^{Ten} automatically recreates a proper Macintosh file, with both a Data Fork and a Resource Fork. Thus, non-MUMM files accessed by applications such as *rcp(1)*, *ftp(1)* and *nfs(4)* that have access to remote non-Mach^{Ten} file systems will be stored in AppleSingle format on the remote file system. When retrieved, the AppleSingle files are reconstituted into a Macintosh file on the local file system.

6.11.3 Utilities for Manipulating Macintosh Files

6.11.3.1 dfork and rfork

Mach^{Ten} provides a UNIX filter — *dfork(1)* — which reads an AppleSingle format file from its standard input and writes only the data fork to the standard output. Mach^{Ten} also provides a filter — *rfork(1)* — which reads an AppleSingle format file from its standard input and writes only the Resource Fork to the standard output.

6.11.3.2 chtype and chcreat

Mach^{Ten} also provides two very useful UNIX utilities for manipulating the type and creator of a Macintosh file. The utilities *chtype(1)* and *chcreat(1)* take a file type or creator, respectively, as the first parameter and a list of files to apply this type or creator to (similar to the UNIX utilities *chmod(8)* and *chown(1)*). Note that using these utilities does not change the data in either fork of a file in any way, it only changes the file’s type or creator.

6.12 Text File Manipulation

Mach^{Ten} gives you the capability to integrate UNIX-created ASCII text files into documents that have been created by Macintosh word processor programs, such as WriteNow or Microsoft Word, and to export Macintosh-created text to UNIX applications.

6.12.1 Combining Macintosh and UNIX Text

Since the Macintosh and UNIX text file formats differ in the choice of line terminator, it may in some cases be necessary to translate the data from one format to the other. Mach^{Ten} provides several mechanisms to convert text files from the Macintosh format to the UNIX format. Choosing the mechanism that works best depends upon the particular situation. Factors such as how the text is created, how it is most often accessed, and how it is archived may influence your decision on which tools work best for you.

6.12.2 Text File Manipulation Utilities

6.12.2.1 mactext

Mach^{Ten} provides a common text file format which permits UNIX text files to be successfully accessed by both UNIX and Macintosh applications. These files have the creator ‘MUMM’ and type ‘TEXT’ and are referred to as MUMM / TEXT files. These files are stored on the file system in the Macintosh text file format, so Macintosh applications can access them normally. Mach^{Ten} automatically translates MUMM / TEXT files to the UNIX text file format when they are accessed by UNIX applications. The *mactext(1)* utility program will create MUMM / TEXT files from traditional UNIX files (type 'BINA', 'nl' termination). The resultant file will be recognizable by UNIX and Macintosh applications.

6.12.2.2 unixtext

The *unixtext(1)* utility program will convert a standard Macintosh text file (type 'TEXT') with 'cr' character line termination to a standard UNIX text file (MUMM / BINA) with 'nl' character line termination. The resultant file will not be accessible by Macintosh word processors. This command is useful when copying a Macintosh text file to a remote non-Mach^{Ten} UNIX system for access by the remote system.

6.12.2.3 Unix <-> Text

The *Unix <-> Text* application in the *Utilities* folder on the Mach^{Ten} CD performs the functions of *mactext* and *unixtext* using a Macintosh user interface. You can use the Finder to drag MUMM / BINA files on top of the *Unix <-> Text* icon to create Mach^{Ten} common text file format files or to convert Macintosh text files to MUMM / BINA files. *Unix <-> Text* only works on Macintosh files that have been saved in "Text Only" mode. In most applications this is done by choosing "Save As ..." from the File menu.

6.12.2.4 dfork.text

Mach^{Ten} provides the UNIX utility program *dfork.text(1)* which extracts the data fork from its standard input (typically a Macintosh text file), translates the 'cr' character to 'nl', and produces the results on its standard output. This utility is useful if you want to leave the original Macintosh file untouched and make a copy of its text for access by some UNIX utilities. For example, to search for a specific word in a Macintosh text file using the UNIX utility *grep(1)*, try:

```
      dfork.text <Macintosh file> | grep <pattern>
```

6.12.3 Text Editing

UNIX provides a number of text editing programs — *vi(1), ex(1), ed(1).* Another popular UNIX editor is GNU *Emacs*. *Emacs* has been ported to Mach^{Ten} and is available on the CD-ROM in the *ported apps* folder. UNIX text editing programs will operate correctly on all Mach^{Ten} text files of type 'BINA', as well as files in the Mach^{Ten} common text format (MUMM / TEXT). They will not recognize Macintosh TEXT files due to the line termination differences described above.

6.12.4 BBEdit

Power Mach^{Ten} includes *BBEdit*, a Macintosh text file editor, which is a complete distribution of the commercial version of *BBEdit* containing numerous new features and accelerated for the PowerPC.

BBEdit has the advantage of recognizing all text file formats without regard for Macintosh or UNIX end-of-line termination characters, making use of the text file manipulation utilities unnecessary. *BBEdit* is recommended for editing UNIX configuration files. *BBEdit* and accompanying documentation and installation instructions can be found in the *Utilities* folder located on the Mach^{Ten} CD-ROM. *BBEdit* includes an HTML (HyperText Markup Language) editor for creating World-Wide Web (WWW) pages.

6.12.5 UNIX to Macintosh "Copy and Paste"

Using conventional Macintosh "copy-and-paste" techniques, you can copy text from a UNIX terminal screen directly to a Macintosh application.

6.13 File Maintenance

Since Mach^{Ten} relies on the Macintosh file system, file maintenance mainly consists of being certain that temporary UNIX working files do not unnecessarily fill up valuable disk space. When running any UNIX system, the percentage of disk space used under normal operation can increase until all of the allocated space is used up. When the file system becomes full, processes run very slowly, if at all, and the system is kept busy notifying the user that the file system is full. The main administrative role, with respect to the file system, is keeping the file system under control.

The following techniques are available to keep the file system manageable:

- monitoring the disk space with *df(1)*
- monitoring files or directories that grow; temporary files, i.e. files in */tmp*, should be purged periodically (*/tmp* is used by numerous executables to store intermediate results)
- identifying and removing inactive files
- identifying users that routinely consume a large amount of disk space.

On a single user system, you will monitor your own usage.

By default, */etc/rc* contains commands to preserve editor temporary files before removing files from */tmp* during system startup.

The following directories and files should be watched for growth:

```
/tmp
/etc/wtmp    (if created by the user)
/var/log/syslog
/var/adm/messages
```

The system administrator may need to create some directories that log administrative messages. If administrative files do not already exist, simply type:

```
touch <filename>
```

to create and timestamp a zero length file.

6.14 Archiving and Backing Up File Systems

Backing up and archiving are two methods of file system protection. In backing up, a complete copy of the system is made on a backup medium. The backup medium can be another system’s hard disk or tape. You can back up your Macintosh files by storing them on a file server, a tape cartridge either locally or on another UNIX machine, or on Macintosh floppies. Archiving usually involves storage of specific files. Large, infrequently used files can be removed and restored when they are needed. Backups should be done on a regular basis to prevent losses due to disk failure or human error.

6.14.1 Archiving to NFS

The following command illustrates how to back up a Macintosh System Folder on an NFS server:

```
tar cf - “System Folder” | compress > /home/server/backup/host.tar.Z
```

where: `tar` is a **t**ape **ar**chive UNIX utility that takes files and directories of files and creates a single, large file. *tar(1)* is used to back up directories on tape and also to combine files for easier electronic file transfer. The directory *home/server* is mount point for the NFS directory storage. *compress(1)* is used to reduce the amount of storage required.

To recover a particular file type:

```
zcat /home/server/backup/host.tar.Z | tar xvfpB - /”System Folder”/file
```

6.14.2 Tape Backups

Mach^{Ten} supports DAT (4mm) and Exabyte (8mm) tape drives. The Mach^{Ten} tape interface has been designed to support a generic SCSI tape device and has been tested specifically using the following mechanisms:

DAT	Models using the Archive Python tape drive (eg. "ClubMac" and "Wang DAT 2600")
Exabyte	Model EXB-8200

Mach^{Ten} can be configured to simultaneously support up to four tape drives, and is configured by default for a single tape device (e.g., 'MAKEDEV mt0' in the following example has already been done). Additional tape device files may be created by executing the following commands (see *mtio(4)* and */dev/MAKEDEV*):

```
# cd /dev
# ./MAKEDEV mt1
# ./MAKEDEV mt2
# ./MAKEDEV mt3
```

Mach^{Ten} uses four device files for each tape drive. For example, the following device files apply to the first tape drive, unit 0. Mach^{Ten} uses a scan of the SCSI chain (from ID-6 to ID-0) to identify tape subsystems and associate them with unit numbers. The first tape drive found is unit 0, the second is unit 1, and so on.

```
/dev/rmt0       -- unit 0, rewind on close
/dev/rmt4       -- unit 0, no rewind on close
/dev/rmt8       -- unit 0, compressed, rewind on close
/dev/rmt12      -- unit 0, compressed, no rewind on close
```

The minor device number encodes two attributes — whether or not to rewind the tape on completion, and whether or not to use compression (if supported by the tape hardware).

The tape device files may be used to read and write tapes using *tar*, *dd*, and other utilities. *Tar* may be used to back up local Mac volumes and / or remote (AppleShare and / or NFS) volumes. Macintosh files will be written in AppleSingle format to preserve both the resource and data forks of the file.

An additional utility *mt* is provided with Mach^{Ten} to position, rewind, and off-load tapes. It may also be used to print out tape information (see *mt(1)*).

Example 1:

Example of archiving your entire Mach^{Ten} file system to tape using a blocking factor of 64, hardware compression, and rewinding the tape when the backup is finished:

```
# tar cvfb /dev/rmt8 64 /
```

Example 2:

Example of archiving two different volumes in two differents save sets on a single tape using a blocking factor of 64, hardware compression, and then rewinding the tape when the second archive is finished.

```
# tar cvfb /dev/rmt12 64 /volumes/MacDisk1
# mt -f /dev/rmt12 eof 1
# tar cvfb /dev/rmt12 64 /volumes/MacDisk2
# mt -f /dev/rmt8 eof 1
```

Note that the *mt* utility is used to place an End Of File marker on the tape after each save set. Doing this enables the *mt* utility to find any of the save sets by searching for the EOF marker from the beginning of the tape. For example, use the following commands to find the start of the second save set in the above example, and then restore */volumes/MacDisk2*:

```
# mt -f /dev/rmt12 fsf 1
# tar xvfbp /dev/rmt12 64
```

7.0 The Mach^{Ten} Network File System (NFS)

The Mach^{Ten} NFS is derived from the Berkeley 4.3BSD Reno release. It is fully interoperable with the NFS shipped by Sun and its licensees. Properly configured, a Macintosh running Mach^{Ten} can share files with mainframes and workstations that support NFS. NFS is an open interface standard that exists on a wide variety of UNIX and non-UNIX PCs, workstations, minis, and mainframes. Mach^{Ten} extends NFS to include the entire Macintosh family.

Mach^{Ten} systems may be used as NFS clients or NFS servers. As a client, a system running Mach^{Ten} can use files stored remotely on NFS networked machines as though they were local files; as a server, a Mach^{Ten} system can store files for other NFS clients. If two Mach^{Ten} servers exist in a single environment, they can be set up to be each other's clients. The Mach^{Ten} NFS capability allows users to cluster diskless Macintoshes and to use a single, large disk on a remote system to provide file service for the cluster. Providing file service for Macintosh systems not only provides for centralized backup and maintenance of disk data, but also minimizes the amount of disk space required to store Macintosh application programs since users no longer need their own individual copy of an application. Moreover, NFS eliminates file copying, which often results in multiple, potentially inconsistent, copies of a particular file, and reduces the need to telnet to specific hosts to gain access to a resource. With NFS, multiple Macintoshes, workstations, and other PCs can give the appearance of having a single, large file system.

Aside from the convenience and economy of file sharing, NFS may provide better overall performance at a lower cost. A server with a large, high performance disk may provide better service than an environment where each PC has its own small disk. Widely-used read-only files can be stored on several servers at the same time, and a complete file system can be distributed across several servers to turn a network into a multiprocessor environment.

7.1 Using NFS

Mach^{Ten} supports remote file system access over Ethernet, AppleTalk, TokenRing, and SLIP and PPP connections. Mach^{Ten}'s extended file system is based on an industry standard developed by Sun Microsystems for file sharing and exchange called NFS® (Network File System). A Mach^{Ten} system can be set up as either an NFS client or an NFS server, or both. This section describes the Mach^{Ten} Network File System installation, operation and maintenance.

7.2 How NFS Works

The Mach^{Ten} NFS performs two major functions — exporting and mounting. In each NFS server, the */etc/exports* file lists the directories that the server will permit others to access. NFS security is controlled by the */etc/exports* file; only file systems that are listed in this file are accessible over NFS. Access to exported file systems can be limited to certain hosts on your network or made available to all other hosts.

Clients may access files on the server by mounting the server's exported directories. A Mach^{Ten} client can mount a directory when the system is launched, or the superuser can explicitly mount and unmount directories by issuing *mount(8)* and *umount(8)* commands. When a client mounts a directory, it does NOT make a copy of that directory. Instead, the NFS software mounting mechanism invokes a series of remote procedure calls to enable the client to transparently access the directories on the server's disk. When a file system is mounted by an NFS client, it appears in the client's file system as does a local file system. NFS makes all files within an NFS system appear as a single file system.

The NFS protocols use Remote Procedure Calls (RPC's) and User Datagram Protocol (UDP) transfers to provide transparent access to users of the client machine. Each time the client needs data from a mounted directory, the client NFS software makes a remote procedure call to the server. Because NFS uses RPC calls that operate over UDP, NFS is "stateless". This means that the server can crash and recover without affecting the client at all, other than a delayed response to the client's request. If a client "mounts" a directory, the *mount* command will be re-sent until the client receives a response from the server. Once the server responds, the client can assume that the information is on stable store and available for the client's use.

NFS servers are supported by programs or daemons that read the special NFS system files, such as the */etc/exports* file. The *portmap* daemon (*portmap(8)*), the *mount* daemon (*mountd(8)*) ,and the NFS daemon (*nfsd(8)*) are started at boot time. The *portmap* daemon provides RPC clients (like *mount*) UDP port numbers for access to the services provided by *mountd* and *nfsd*. The *nfsd* daemon listens for service requests on the NFS port as specified in the "Network File System Protocol Specification, RFC 1094". The *mountd* daemon determines whether or not the client issuing the *mount* request has access rights to the requested directories. If the client has the appropriate permissions, the server permits access to the file system. When a directory is mounted it is not transferred to the client machine, but instead appears in the client's directory structure as though it were locally stored. Once a directory is mounted, the client accesses remote files in the same manner it accesses local files. Standard UNIX directory and file manipulation commands can be applied to remote directories in exactly the same way they are applied to local directories.

The following man pages describe all of the NFS daemons, commands, and special files: *exports(5), fstab(5), inetd(8), mount(8), mountd(8), nfsd(8), portmap(8), nfsiod(8)* and *rpcinfo(8)*. You should have a general familiarity with the names and functions of these network file system components.

7.3 Setting Up an NFS Server or an NFS Client

Every Mach^{Ten} system can be set up as an NFS client, an NFS server, or as a system that acts both as a client and a server. An NFS client system receives services from an NFS server; an NFS server provides services to one or more NFS clients. A client may or may not have a disk; a server must have at least one local disk. A client mounts directories, while a server exports directories. Clients play an active role, initiating a binding between themselves and a server; the server completes the binding. The server plays a passive role, simply advertising files that clients can access; the clients actively mount directories that are advertised by the server.

7.3.1 Server Exporting

The server advertises the directories that it is prepared to export in the */etc/exports* file. Each directory entry also contains any access restrictions that may apply. This file is created by the system administrator using a text editor, and processed by *mountd* upon startup.

An */etc/exports* file has the following syntax:

```
      directory  -option[, option]  hostlist
```

where, *directory* is the path name of a directory, *option* is a list of options, and *hostlist* is a space separated list of host names indicating who can access the directory. The following is a typical */etc/exports* file. Note that there must not be any comments or blank lines in this file:

```
      /
      /usr  -ro                     payrollclient accountingclient
      /etc
      /Applications
      /exports/root/gladys          gladys
```

If an */etc/exports* file exists, Mach^{Ten} will automatically launch *portmap*, *mountd*, and *nfsd* at boot time via the */etc/rc* file. Otherwise, the NFS server daemons will not be started.

7.3.2 Client Mounting

Clients may gain access to advertised service files and directories by mounting that file or directory. Directories and files are mounted either by an explicit *mount* command issued by the client, or automatically at system startup by the */etc/rc* file. If a client needs a particular directory for a limited amount of time, the *mount* and *umount* commands can be used as needed. A client can mount any directory that is in the server's */etc/exports* file as long as that client has access permission. Before a client can issue a *mount*, the client must prepare a mount-point — that is, allocate a place for the exported file or directory. Clients should use the *mkdir(1)* command to make a directory for each directory the client wishes to mount.

The *mount* command has the following syntax shown:

```
      /usr/etc/mount -t type [-rw] -o [options] server:/dir /mount_point
```

where,

t	type of mount (type is "nfs" for nfs mounting)
r	read-only
w	read-write

o	indicates that a list of options follows; e.g., soft (try once) or hard (keep trying)
server	name of the server
/ dir	name of the exported directory
/ mount_point	client mount_point

Refer to the *mount(8)* manual page for more details.

Diskless clients can be configured so that directories that are needed all of the time are mounted automatically when the system is booted. The command:

```
mount -a
```

will instruct *mount* to look in the */etc/fstab* file for a list of the file systems to be mounted. The syntax for entries in the *fstab* file is as follows:

```
server:server_dir client_mount-point type options freq pass
```

where,

server	name of ther server
server_dir	the directory that the client wishes to mount
client_mount-point	the directory that the client uses to access *server_dir*
type	nfs (for remote NFS mount)
options	(mounting options)
freq	0 (needs to be included, but not used in Mach^{Ten})
pass	0 (needs to be included, but not relevant to Mach^{Ten})

As an example, the entry:

```
omar:/space    /mnt    nfs    rw,soft    0    0
```

describes the directory */space* on a remote host named *omar* and a mounting point on the local system of */mnt*. The directory is an NFS directory. When mounted, the user will have read and write access to the directory. *Mount* will return an error if the server is not responding.

At boot time, Mach^{Ten} will automatically mount the file systems listed in */etc/fstab* if the MOUNT_REMOTE_FILES variable is set in the file */etc/rc.conf*. This file is used by */etc/rc* at start up (see *rc(8)*).

7.3.3 Set-Up Summary

Note that only the NFS server needs to have the NFS daemons running. The server needs an */etc/exports* file and the client needs to either explicitly mount file systems delineated in the *exports* file, or list the directories that the client wishes to mount in the */etc/fstab* file and set the MOUNT_REMOTE_FILES variable in the */etc/rc.conf* file.

Configure Mach^{Ten} as an NFS server and client following the steps below:

1. Find out the host names and IP addresses of the machines that will act as NFS servers on your network. You can configure a Mach^{Ten} host as an NFS server or use an existing NFS server on your network (a Sun workstation, for example).

2. Find out the host names and IP addresses of machines on your network that will receive NFS service. Unless you have an internet domain name server on your network, the server needs to have all of its clients listed in its */etc/hosts* file. Likewise, each client needs to have the address of its NFS server listed in its */etc/hosts* file (unless the name server is in use).

3. Become superuser so that you can edit the */etc/exports* and the */etc/fstab* files.

4. Edit the */etc/exports* file on the server so that it includes each directory and file that will need to be exported to clients. Make sure that each entry includes the proper restrictions. If you do not intend for your system to act an an NFS server, skip this step. If your system is to be used as an NFS server, restart Mach^{Ten} now to bring up the NFS server daemons.

5. Edit the */etc/fstab* file on each client system. Create an entry in the file for each directory to be mounted.

6. Create the appropriate directories to be used as mount points listed in */etc/fstab* on each client machine using the *mkdir(1)* command.

7. Use the command *umount -a* to unmount all directories that are not busy, that is those directories that are not in the user's working directories. This unmounts directories previously mounted, for example, by an older */etc/fstab* file.

8. Use the command *mount -a* to mount everything in the current */etc/fstab* file. Each time the client system is booted up, if */etc/rc.conf* has MOUNT_REMOTE_FILES set, a *mount -a* command is performed automatically.

7.4 NFS Volumes on the Macintosh Desktop — Desktop NFS

Mach^{Ten} extends the traditional NFS service (accessible from UNIX applications) to include access to NFS volumes from Macintosh applications. This feature is called Desktop NFS because the NFS volumes appear on the Finder’s desktop. Using Desktop NFS, remote file system data is made available to local Macintosh applications by representing the NFS server as a volume icon on the Macintosh desktop. Macintosh application references to folders and files in these volumes are intercepted by Mach^{Ten} and translated into requests for foreign NFS file access. Desktop NFS has been designed to operate as much as possible like Apple File Sharing.

Desktop NFS provides support for both AppleSingle and AppleDouble file formats.† To make an NFS volume visible on the desktop, use the *dtmount* command after mounting the NFS volume. To make the directory *Marketing* from the server *Opus* visible on the desktop as the volume named *Shared*, use the following command:

```
mount -t nfs -o rw,soft Opus:/Marketing /Shared
dtmount /Shared
```

Note that the path */Shared* must exist on the Mach^{Ten} system. The name *Shared* will be the Macintosh Volume name assigned to the NFS volume. It will appear under the NFS Server Icon on the Finder desktop. Desktop NFS mounts can also be entered in the */etc/fstab* file (see *fstab(5)*).

Mach^{Ten} *dtnfs* automatically recognizes and performs the proper file system text translations for Macintosh applications. This transparent file translation enables Macintosh applications to operate on UNIX text files residing on a Desktop NFS volume.

Mach^{Ten} *dtnfs* optionally supports Macintosh file and record locking via *bwnfsd* or *pcnfsd*, which are locking daemons that run on the NFS server. Note that Mach^{Ten}’s NFS server provides *bwnfsd(8)*, but not *pcnfsd*.

7.4.1 Desktop NFS Volume Mounting

Mach^{Ten} *dtnfs* is in essence an NFS client with special capabilities. The previous section describing NFS server and client setup apply to *dtnfs* as well. *Dtnfs* is activated when the *dtmount* command is issued. As an example, the command:

```
mount -t nfs -o rw,soft omar:/space /mnt
```

will mount the */space* file system from a remote NFS server named *omar* on the local client Mach^{Ten} system at mount point */mnt*. The command:

```
dtmount /mnt
```

† Copies of the “AppleSingle / AppleDouble Formats for Foreign Files Developer’s Note” may be obtained from the Developer Technical Publications division of Apple Computer, Inc. (1-800 / 282-ADPA).

promotes the NFS file system to appear as a network volume icon named *mnt* on your local desktop. Traditional Macintosh point and click operations using the volume icon will generate NFS requests on *omar*'s */space* directory while having the appearance to the Finder and other Macintosh applications that the volume and all files and folders within it are on an AppleShare server.

To remove the *dtnfs* volume from the desktop, drag it to the Trash. This will sever the linkage between the desktop and the remote NFS server. If you prefer, *udtmount(8)* may be used to remove the volume:

```
udtmount /mnt
```

Finally, to completely remove the NFS Server from the NFS Client, use the *umount(8)* command.

```
umount /mnt
```

7.4.1.1 Automatic Mounting of Desktop NFS Volumes

As with non-*dtnfs* volumes, the */etc/fstab* file is searched for *dtnfs* volumes to mount in response to a *mount -a* command. A *dtnfs* entry is distinguished from a non-*dtnfs* entry in */etc/fstab* by the option *dtnfs*:

```
omar:/space    /mnt     nfs      dtnfs,rw,soft     0    0
```

7.4.2 Macintosh Record Locking on Desktop NFS Volumes

A *dtnfs* mounted volume will advertise support for Macintosh record and file locking to Macintosh applications if a special option is provided in the *mount* command or in the */etc/fstab* entry for the *dtnfs* file system. With *dtnfs* record locking enabled, Macintosh applications that make use of record locking on AppleShare volumes will also support record locking on *dtnfs* volumes.

Dtnfs record locking works in conjunction with a locking daemon running on the remote NFS server. For Mach^{Ten} NFS servers, the locking daemon is *bwnfsd(8)* and the option flag supplied to mount (or placed in */etc/fstab*) to enable locking is *bw_lock*:

```
mount -t nfs -o dtnfs,bw_lock,rw,soft omar:/space /mnt
```

For Sun NFS servers, the locking daemon is *lockd*, and the option flag supplied to mount is *lockd*:

```
mount -t nfs -o dtnfs,lockd,rw,soft omar:/space /mnt
```

The locking daemon must be running on the NFS server when the desktop NFS volume is mounted. Under Mach^{Ten}, the command:

```
/usr/etc/bwnfsd -A
```

starts the locking daemon. Place this command in the file */etc/rc* to automatically start the locking daemon on system start up.

7.4.2.1 Bwnfsd UID and GID Mapping

Bwnfsd also provides a mapping between numeric user and group IDs to their corresponding symbolic names. *Dtnfs* uses this service of *bwnfsd*, by default, if it finds a running *bwnfsd(8)* daemon on the NFS server. It is not an error if a *bwnfsd* daemon is not found on the NFS server AND the only service desired of *bwnfsd* is the UID and GID mapping. In this case, UIDs and GIDs are mapped to the strings *user %d* and *group %d*, where *%d* represents the numeric UID or GID. To disable this default use of the *bwnfsd*'s mapping services, use:

```
dtmount -o nobw_map /SharedDisk
```

The option -*A dirpath* selects the spooling area for printing requests.

7.4.3 Authentication Options

Desktop NFS requires the specification of a user's credentials when an NFS server is being promoted to the desktop. Once these credentials are authenticated, they will be used for all accesses made of the NFS server caused by the Macintosh applications which use this DTNFS volume. The Desktop NFS credentials can be specified in a number of ways, and may be authenticated using either the local */etc/passwd* database, or the server's */etc/passwd* database, via *bwnfsd(8)* or *lockd*.

The mechanisms to mount DTNFS volumes are described below. In each case, it is assumed that the NFS Server *opus:/Marketing* has already been mounted on the path */SharedDisk* by the root user for access by the UNIX processes. Typically this is done by listing the NFS Servers in the */etc/fstab* file, and then */etc/rc* will automatically mount these file systems.

7.4.3.1 Credentials of Current User

In the traditional case, it is desirable to use the credentials of the user who mounts the DTNFS volume. In this case, the user has already been authenticated at login time, and has been assigned a valid set of credentials. To use these credentials for the DTNFS volume, use:

```
dtmount /SharedDisk
```

The credentials used will be those of the user who issued the *dtmount* command.

7.4.3.2 Prompt for Username and Password

To closely mimic the Apple File Sharing model, a user may be prompted for a username and a password for each DTNFS volume added to the desktop. This username and password is authenticated by the server and, if the authentication succeeds, the server returns the credentials to be used when accessing this volume. To prompt the user for a username and password, use:

```
dtmount -o bw_auth  /SharedDisk
```

This command will display a Macintosh style dialog box requesting a username and a password. Once the username and password are entered, they are passed to the server for authentication. In this case, *bwnfsd(8)* is used to do the authentication, so a *bwnfsd* daemon must be running on the server. If the authentication succeeds, the volume will appear on the desktop and the credentials used will be those returned by *bwnfsd*.

7.4.3.3 Username and Password on Command Line

It is also possible to specify the username and password on the *dtmount* command line. This situation could be used if an administrator was setting up a Mach^{Ten} system for a particular user, and it was desirable to mount the DTNFS volumes automatically, without any interaction from the user. To specify a username and password on the command line, use:

```
dtmount -o bw_auth2,username,password  /SharedDisk
```

This command passes the given username and password to the server for authentication. In this case, *bwnfsd* is used to do the authentication, so a *bwnfsd* daemon must be running on the server. If the authentication succeeds, the volume will appear on the desktop and the credentials used will be those returned by *bwnfsd*.

7.4.3.4 User ID and Password on Command Line

It is also possible to specify the user ID and password on the *dtmount* command line. This situation could be used if an administrator was setting up a Mach^{Ten} system for a particular user, and it was desirable to mount the DTNFS volumes automatically, without any interaction from the user. To specify a user ID and password on the command line, use:

```
dtmount -o bw_auth3,userid,password /SharedDisk
```

This command passes the given user ID and password to the server for authentication. In this case, *bwnfsd* is used to do the authentication, so a *bwnfsd* daemon must be running on the server. If the authentication succeeds, the volume will appear on the desktop and the credentials used will be those returned by *bwnfsd*.

7.4.4 File System Security

As the sharing of file system data becomes more prevalent, the possibility of external tampering or damage to local file data exists. Mach^{Ten} enforces UNIX user name and group file system security for all exported NFS files and folders. Each remote NFS request contains validated user and group identifiers which determine whether a file or directory / folder can be accessed or modified. The Mach^{Ten} *chown* and *chmod* commands can be used to set ownership and access rights for both UNIX and Macintosh files. These rights will be enforced by the NFS server.

7.4.4.1 AppleShare Access Privileges

AppleShare access privileges are imposed on each folder on an AppleShare server. These privileges apply to the files and folders within a folder. They also apply to the index of a folder. (The index of a folder is the list of entries within the folder, i.e., a list of files and other folders.)

The access privileges for AppleShare folders are:

See Files	Permission to read the index of a folder. Permission to read the files listed in the index.
See Folders	Permission to read the index of a folder. Permission to see the folders listed in the index.
Make Changes	Permission to read the index of a folder. Combined with “See Files” or “See Folders”, provides permission to add or delete index entries for files or folders, respectively. Combined with “See Files”, provides permission to make changes to the files listed in the index. Permissions to make changes to the folders listed in the index are subject to the access privileges of each of those folders.

7.4.4.2 UNIX Access Privileges

UNIX access privileges are imposed on each file and directory on a file system.

The access privileges for UNIX files are:

Read	Permission to read the file.
Write	Permission to write the file.
Execute	Permission to execute the file (the file is a program).

The access privileges for UNIX directories are:

Read	Permission to read the index of a directory.
Write	Permission to make changes to the index of a directory.
Execute	Permission to locate a directory entry, given the entry is in the index. In UNIX, there is a separation of the actions of finding an entry in the index and actually locating this entry on the file system. The permission to locate the entry is the “Search” permission.

Note that in UNIX, permissions to read or write the files within a directory are not granted by the permissions of the directory, but rather by the permissions of the files themselves. Also, the permissions to “Read”, “Write” or “Search” the index of a directory does not differentiate between files and directories in the index.

7.4.4.3 Mapping AppleShare Access Privileges into UNIX Access Privileges

The default Apple File Sharing permissions allow all other users and groups to access newly created files and folders. Desktop NFS mimics this behaviour by assigning the access privilege of “Read”, “Write”, and “Execute” for each of the owner, group, and others when a Macintosh application creates a file or folder on a Desktop NFS volume. Since the Apple File Sharing default is not settable on an individual user’s basis, Desktop NFS does not use a *umask* to specify the permissions to be used when creating a new file or folder on a Desktop NFS volume. The Finder’s Sharing Dialog Box can subsequently be used to restrict access to such newly created folders. Note that the Apple File Sharing permissions apply only to folders. To set the access privileges of individual files, the UNIX interface must be used. Since NFS supports permissions for both files and folders, Desktop NFS users can further restrict access to Desktop NFS files by specifying the access permissions for the individual files themselves. In this case, the UNIX *chmod* command must be used, as the Finder’s Sharing Dialog Box will only allow making changes to folders.

Since the UNIX access permission to “Read”, “Write” or “Search” the index of a directory does not differentiate between files and directories in the index, Desktop NFS does not allow setting one of “See Files” and “See Folders” in the access privileges of a folder. Either both are permitted or both are not. A Desktop NFS folder with “See Files” and “See Folders” permitted actually has the UNIX permissions “Read” and “Search”. A Desktop NFS folder without “See Files” and “See Folders” permitted does not have the UNIX permission “Read” and may not have the UNIX permission “Search”, depending on the setting of the “Make Changes” permission.

A Desktop NFS folder with “Make Changes” permitted actually has the UNIX permissions “Write” and “Search”. A Desktop NFS folder without “Make Changes” permitted does not have the UNIX permission “Write” and may not have the UNIX permission “Search”, depending on the setting of the “See Files” and “See Folders” permissions (as above).

7.4.4.4 Access Privilege Strategies

Locked Folders. A Desktop NFS folder without “See Files”, “See Folders”, and “Make Changes” permissions has none of the UNIX “Read”, “Write”, or “Search” permissions. This mode is used to maintain complete privacy on a folder and is called a locked folder. Desktop NFS locked folders are as secure as AppleShare locked folders and require no additional permission settings on the individual files within the folder.

Bulletin Board Folders. A Desktop NFS folder with “See Files” and “See Folders” permissions but without “Make Changes” permissions has the UNIX permissions “Read” and “Search”. This mode is used to maintain a bulletin board style of privacy — users can open and read the files but are prevented from changing or deleting them. In Desktop NFS, the permissions to “Read” or “Write” the files within a bulletin board folder are granted by the permissions of the actual files themselves. To implement a bulletin board folder in Desktop NFS the files should have only the UNIX permission “Read”.

Shared Folders. A Desktop NFS folder with “See Files”, “See Folders”, and “Make Changes” permissions has the UNIX permissions “Read”, “Write”, and “Search”. This mode is used to maintain a shared folder — users can open, read, change, add, and delete files. In Desktop NFS the permissions to “Read” or “Write” the files within a shared folder are granted by the permissions of the actual files themselves. To implement a shared folder in Desktop NFS the files should have the UNIX permissions “Read” and “Write”.

Note that with Desktop NFS, it is possible to mix bulletin board and shared folder styles, with each file maintaining its own privileges.

Drop Folders. A Desktop NFS folder without “See Files” and “See Folders”, but with “Make Changes” permissions, has the UNIX permissions “Write” and “Search”. This mode is used to maintain a drop folder. Users can deposit files or folders but cannot read, remove, or change anything in the folder.

7.4.4.5 Differences Between Apple File Sharing and DTNFS

In Apple File Sharing, users can give away ownership of the files they own. More specifically, users can change the owner of a file from themselves to some other user. In NFS, only the superuser (or root user) can change the ownership of a file. Therefore, with DTNFS it is not possible for a non-root user to change the owner of a file. Attempting to do so results in an error.

7.4.5 AppleDouble

Desktop NFS supports two formats for storing Macintosh files on NFS servers. The default format is AppleSingle. In this format, each file contains an AppleSingle header, a Data fork, and a Resource fork. AppleSingle files autonomously contain all of the necessary Macintosh information in a single file. The other supported format is AppleDouble. In this format, the Data fork of the Macintosh file is contained in one file while the AppleDouble header and the Resource fork are contained in another file. The name for the file containing the AppleDouble header and the Resource fork is created by appending the percent '%' character to the beginning of the original files name.

The AppleDouble format provides convenient sharing of data between Macintosh applications and their counterparts that run on other platforms (like IBM PC or UNIX workstations). It is necessary for the cross platform applications to share a common format for the data passed between platforms.

A Mach^{Ten} Desktop NFS client can simultaneously read files in both the AppleSingle or AppleDouble format. To specify the format to be used when creating a file on a Desktop NFS volume, use the double option when promoting the NFS volume to the desktop.

To use the AppleDouble format when creating files use:

```
dtmount -o double /SharedDisk
```

To use the default AppleSingle format when creating files, do not specify the double option, use:

```
dtmount /SharedDisk
```

7.4.6 DOS Extensions

Desktop NFS maps files with DOS extensions (trailers) to configurable Macintosh creators and types. These mappings are specified in the *Dext* resource in the Mach^{Ten} application. This resource may be edited using *ResEdit*. It has its own template, with the following fields::

DOS Trailer	DOS extension, including the dot '.' (e.g., .XLS)
Creator	Macintosh Creator (e.g., XCEL)
Type	Macintosh Type (e.g., XLS3)

Some special cases can be specified in the *Dext* resource to specify default mappings for files with no DOS extension, or DOS extensions that do not match any of the specified extensions.:

before	Before checking for AppleSingle or AppleDouble headers, and before looking at data in the file, map it to this creator and type. This option is intended for use on NFS servers with slow access times and possibly hundreds of files in each directory (e.g., An optical Juke Box). In this case, checking each file for AppleSingle / Double headers or checking to see if it contains text may take a long time. Setting the *before* special case prevents such checking, and immediately assigns a default creator and type to the file.

text	After checking for AppleSingle and AppleDouble format, and after looking at the data in the file, if data is text, map it to this creator and type.
binary	After checking for AppleSingle and AppleDouble format, and after looking at the data in the file, if data is not text, map it to this creator and type.

The mapping algorithm to assign Creators and Types to files on DTNFS volumes follows:

1. Does the file have a DOS extension, and does this extension match any of the specified extensions? If so, use this extension's specified Creator and Type.
2. Is the special case specification before set? If so, use the given Creator and Type.
3. Does the file have an appropriate AppleDouble pair? If so, use the Creator and Type specified in the AppleDouble header (within the AppleDouble file).
4. Does the file have a valid AppleSingle Header? If so, use the Creator and Type specified in the AppleSingle header.
5. Does the file contain text?
    a. Is the special case specification text set? If so, use the given Creator and Type.
    b. Otherwise use the Creator 'MUMM' and the Type 'TEXT'.
6. The file is a binary UNIX file.
    a. Is the special case specification binary set? If so, use its Creator and Type.
    b. Otherwise use the Creator 'MUMM' and the Type 'BINA'.

8.0 Printing

Mach^{Ten} supports UNIX-style spooled printing to PostScript LaserWriters on AppleTalk, to ASCII-based ImageWriters on serial ports, and to remote UNIX printers via Berkeley's line printer daemon (*lpd(8)*). These printing capabilities support both UNIX and Macintosh applications. This means that with Mach^{Ten} your Macintosh can take full advantage of UNIX-style printer sharing and printer administration.

With this hybrid system, interconnected Mach^{Ten} and other UNIX systems are able to spool text and PostScript to traditional UNIX printers, as well as AppleTalk LaserWriter and ImageWriter printers. Mach^{Ten} supports full UNIX print job control, including the management of multiple printers, multiple spooling queues, and access to printers either local or accessible across a network. In addition, using the laser writer server daemon (*lwsrv*), Mach^{Ten} can emulate a named AppleTalk LaserWriter allowing remote Macintosh applications to direct printer output to a Mach^{Ten} system which will, in turn, spool the output as a Mach^{Ten} print job.

8.1 Mach^{Ten} Print Spooling

Printing in a standard Macintosh environment requires competing for the exclusive use of a printer and waiting for the relatively slow printing process to complete. With Mach^{Ten}, every Macintosh is able to be more efficient in its printing by becoming a spooling client/server system. In a Mach^{Ten} spooled system, text that would normally be sent directly to an AppleTalk LaserWriter is instead directed to a client program. The client transfers the data to a spooling area monitored by a spooling server program and returns control to the application. Thus, applications are only required to spend the amount of time it takes to transfer the print data from one disk to the spooling disk before being able to continue. Since Mach^{Ten} is a multi-tasking system, the server continues to run in the background and feeds the spooled data one file at a time to a designated printer. In a networked printing environment, the server forwards the spooled data to a remote server for printing from the remote system. Thus, Mach^{Ten} provides not only local printer spooling, but remote printer spooling as well.

The client program in the Mach^{Ten} print spooling system is *lpr(1)*. The background server program, also known as the print daemon, is *lpd(8)*. Detailed information on both *lpr* and *lpd* and other programs used by the Mach^{Ten} spooling system can be found in the online manual pages, as well as in the document "4.3BSD Line Printer Spooler Manual" in the System Manager's Manual section on the Mach^{Ten} CD-ROM.

8.1.1 The Print Spooler Database

The */etc/printcap* file (see *printcap(5)*) is the master database for the Mach^{Ten} printing system. It describes all printers directly attached to your system, as well as printers accessible across the network. Mach^{Ten} is shipped with an initial */etc/printcap* file which you may need to modify depending on your printing configuration.

8.2 Local Printing

In order to print with a minimum of system configuration, Mach^{Ten} comes configured for local spooling of print data. With local spooling, a Mach^{Ten} system acts as both print spooling client and server.

Mach^{Ten} supports two local print capabilities for UNIX and Macintosh applications:

- Spooled printing to a PostScript LaserWriter on AppleTalk. This is the default configuration.
- Spooled printing to an ASCII-based ImageWriter or DeskWriter connected directly to a serial port.

8.2.1 Printing Text Files to a PostScript LaserWriter on AppleTalk

To use a LaserWriter on AppleTalk, simply select the printer by using the Chooser Desk Accessory. Then use the *lp(1)* command as follows:

```
lp <file_name>
```

Lp is a Mach^{Ten} shell script which invokes *pstext(1)*, generating formattedPostScript output from the text file. The PostScript output is then queued to the spooling area by *lpr*, where *lpd* spools the file to the LaserWriter on your AppleTalk network selected by Chooser. This is the default configuration for Mach^{Ten}. If you are not already running *lpd*, *lp* will start it for you. *lp* will accept *pstext* options allowing you to control the output format with respect to orientation, headers, fonts, etc.

8.2.2 Spooled Printing to an ASCII-Based ImageWriter Connected Directly to a Serial Port

UNIX files can be printed to an attached ImageWriter or DeskWriter using the UNIX *pr(1)* command by utilizing the ImageWriter’s ASCII text printing capability.

To print to a locally connected ImageWriter, perform the following steps:

1. Edit the */etc/printcap* file and uncomment (remove the pound (#) marks) from the beginning of each line for the ImageWriter entry. The uncommented entry should look like this:

```
iw|lp|ImageWriter|ImageWriter II:\
    :MF:br#9600:fs#06320:\
    :lp=/dev/ttyfa:\
    :sd=/var/spool/iw:\
    :tr=\f\f:\
    :of=/usr/lib/lpf:\
    :af=/var/spool/iw/acct:\
    :lf=/var/spool/iw/errs:
```

NOTE: If you have a locally connected DeskWriter, uncomment the entry starting with `dw|lp|DeskWriter:\` instead. The DeskWriter entry differs from the ImageWriter entry in the setting of the serial port speed (baud rate) designated by the `:br#<table entry>:` field in the entry. Modify this field as necessary to match the baud rate of your DeskWriter according to the following table:

br# entry	**DeskWriter Baud Rate**
50	14.4 Kbps
75	28.8 Kbps
110	57.6 Kbps
300	300 bps
600	600 bps
1200	1200 bps
1800	1800 bps
2400	2400 bps
4800	4800 bps
9600	9600 bps
19200	19.2 Kbps
38400	38.4 Kbps

The default serial port speed setting for DeskWriter entry in */etc/printcap* is 57.6K baud.

2. While still editing */etc/printcap*, if the ImageWriter is connected to the printer port, change the line that says `lp=/dev/ttyfa` to read `lp=/dev/ttyfb`. If the ImageWriter is connected to the modem (phone) port, skip this step — *printcap* is already set in this configuration.

3. Print a UNIX file by using one of the following two commands:

```
pr <file_name> | lpr -Piw
lpr -Piw <file_name>
```

   The first version will paginate the output; the second version will produce "raw" output.

If *lpd* is not running, the following error message will appear:

```
lpr:  connect:  No such file or directory
jobs queued, but cannot start daemon
```

In this case, simply start the daemon by giving the command "lpd".

NOTES: The ImageWriter is not selected with the Chooser, but with the *-P* option of the *lpr* command. The name following *-P* may be any one of the names listed on the first line of the */etc/printcap* entry for the ImageWriter — *iw* and *ImageWriter* work equally well in this case.

Macintosh files, such as Microsoft Word files with embedded formatting characters, must be printed from their Macintosh application and cannot be printed with an *lp*, *lpr* or *pr* command.

Macintosh programs running on the Macintosh can access the ImageWriter as they normally do via Chooser.

8.3 Remote Printing

In a networked printing environment, Mach^{Ten} supports spooling to a remote UNIX or Mach^{Ten} system with an attached printer via *lpd*. The local *lpd* uses the TCP protocol to communicate with the remote *lpd*. Your *MachTen* system can be configured as an *lpd* client (invoking remote printing services) or an *lpd* server (spooling a received file to its local printer). Both ends of the *lpd* connection are configured through line entries in */etc/printcap*.

8.3.1 Spooled Printing to a Remote UNIX Printer

To set up Mach^{Ten} to spool files to a remote printer, perform the following steps:

1. Configure the other machine to spool to the printer, following the instructions for that machine and printer. If the machine is a Mach^{Ten} system, refer to the instructions below. It is possible this step is unnecessary if a machine already exists on your network that is doing spooling. Test that the remote machine can print by printing a document while logged on to that machine.
2. Edit the local */etc/printcap* file and uncomment (remove the pound (#) marks) from the beginning of each line for the remote entry. The uncommented entry should look like this (substitute the printer name you wish to use for `mumm` on line 1 and the remote host machine name on line 2 (`rm=<host>`):

```
mumm: \
:lp=:rp=lp:rm=mumm:sd=/var/spool/mummlpd:lf=/var/adm/ldp-errs:
```

3. In addition to knowing the remote spooler's name and printer, you must also have print access to the remote host. To obtain print access, your host name must be entered into the remote spooler's */etc/hosts.equiv* or */etc/hosts.lpd* file. Enter the full domain name in either file, for example: *myhost.tenon.com*.
4. Create the spool directory */var/spool/mummlpd*. The actual name chosen typically references the remote host name. The directory name must match the `sd=` entry above. Give write permission to all users by using *chmod(1)*.

5. Start the line printer daemon:

```
lpd
```

6. Print:

```
pr /etc/passwd  | lpr -Pmumm
```

NOTE: The name used with the *-P* option corresponds to the first line of the remote printer entry in */etc/printcap*.

8.3.2 Receiving Remote Print Jobs

Mach^{Ten} can act as a print spooler for other UNIX machines on your network. The printer can be a directly-connected ImageWriter, a LaserWriter on AppleTalk, or any printer on the network. To set up Mach^{Ten} to act as a print spooler for other UNIX machines on the network, perform the following steps:

1. Configure your local printer following the instructions in section "8.2 Local Printing" (previously described).
2. To enable remote print access, add any remote host names that will use the printer as a remote printer to the */etc/hosts.equiv* file or the */etc/hosts.lpd* file. Refer to the *rlogin(1)* manual page for more information on the format of the */etc/hosts.equiv* file. Add the host name as a full domain name, for example *myhost.tenon.com*.
3. Start the line printer daemon:

```
lpd
```

4. Enable the printer with the command:

```
lpc enable <printer>
```

8.4 Selecting an Alternate Printer

Printing involves explicit or implicit selection of a physical printer. To explicitly select a printer you may use the *-P* option for the *lpr* command. If no *-P* option is specified, the *lpr* program uses the value of the PRINTER shell environment variable (see *printenv(1)*). If the environment variable does not exist, *lpr* uses the the first printer with the name *lp* specified in the */etc/printcap* file. In the default configuration, this is also the AppleTalk printer selected via Chooser.

For environments with multiple printers on AppleTalk, selecting a printer other than the Chooser printer requires an entry in */etc/printcap*. A valid entry looks like:

```
:at=mylaser\:LaserWriter@*:
```

An at entry in the */etc/printcap* file specifies the AppleTalk name of a printer. An at entry must include the complete AppleTalk name, type, and zone designator in standard AppleTalk format (`<name>:<type>@<zone>`). A '*' may be used to refer to the "local" zone.

The colons at the beginning and end of the entry are required to distinguish the entry from other parameters. The above entry designates a LaserWriter with the AppleTalk name *mylaser*, of type *LaserWriter*, associated with the local zone. Note the use of the required backslash to escape a colon internal to the specification. If the name field entry is null, the Mach^{Ten} system will retrieve the name of the printer from the information saved by the Chooser.

8.5 Extending Remote Printing to Macintosh Applications

The Mach^{Ten} UNIX spooling capability is not only available for UNIX applications, but is extended to all Macintosh applications (even Macintosh applications on non-Mach^{Ten} systems) through a LaserWriter server named *lwsrv(8)*. *Lwsrv* is a background Mach^{Ten} program which identifies itself to the local AppleTalk zone as a LaserWriter. Remote Macintoshes can select this "synthetic LaserWriter" as an output device with their Choosers. Once selected, Macintosh application print operations automatically direct printer output to *lwsrv*. Any Macintosh application on a remote Macintosh can send printer output to an *lwsrv* printer. *Lwsrv*, in turn, passes the printer output to *lpd* and *lpd* spools to the designated printer as per Chooser or the */etc/printcap* entry.

NOTE: Applications running on the same Macintosh as *lwsrv* cannot send output to *lwsrv* due to a lockout condition in the MacOS network software. This is not a problem in environments with more than one Mach^{Ten} system, since Macintosh applications can send their output to an *lwsrv* on another machine for spooling.

As an example, imagine four devices — a printer, a Sun workstation that is the print spooler for the printer, a Mach^{Ten} Macintosh, and Macintoshes not running Mach^{Ten}. In this example the Printer and the Sun are on Ethernet, the Mach^{Ten} Macintosh is on both Ethernet and AppleTalk, and the Macintosh clients are on AppleTalk.

The Macintosh user on Pippin wants to print to the Printer. An *lwsrv* daemon running on Max enables him to do this. The printer will appear in the Choosers of Pippin and Granny Smith. When Pippin and Granny Smith choose "Printer" and print, printer output is sent from Pippin and Granny Smith to the printer via Max and *lwsrv*.

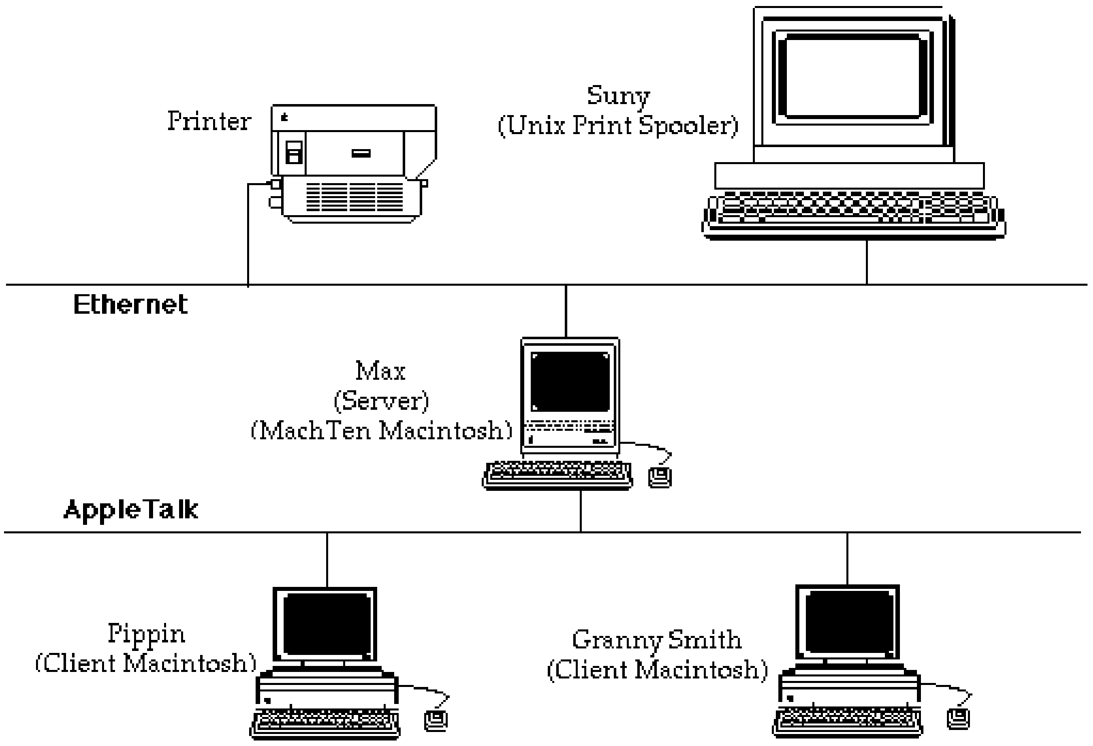


Figure 23. Spooling Printer Output to UNIX Printers Using lwsrv

8.5.1 Lwsrv Configuration Files

Lwsrv requires two types of configuration files. A single file describes the fonts available in the printer being used. The other files contain QuickDraw to PostScript macro that this Macintosh user has on the client Macintoshes (Granny Smith and Pippin in the example). Tenon supplies font files that will work for LaserWriter or LaserWriter Plus printers and a macro (LaserPrep) file that will work for Macintosh users that have installed version 5.2 or 7.0 of LaserPrep under MacOS System 6 or for MacOS System 7 users.

8.5.1.1 Font Lists

When a Macintosh wants to send printer output to a particular printer, it first inquires about available printer fonts. It uses this information to map its internal fonts to the fonts available on the printer. If necessary, it will download any additional fonts needed for the document being processed. In order for *lwsrv* to work correctly, *lwsrv* must know what fonts are available on the printer being used.

The Tenon-supplied font files */usr/lib/LWFonts* and */usr/lib/LWPlusFonts* contain font lists for use with LaserWriter and LaserWriter Plus printers. However, in order to use printers other than a LaserWriter or LaserWriter Plus, the list of fonts must be obtained and placed in a file for *lwsrv* to read at startup time. One way of doing this is to use the Apple LaserWriter font utility which is on the System Software disk. A second way is to run *lwsrv* in trace mode (using the *-t* option — see the *lwsrv(8)* man page). The data obtained must be edited to be in a usable form known as a font dictionary. Under the newer LaserPrep, the font list is encapsulated in a query that can be retrieved by running *lwsrv* with tracing on. The appropriate code is framed with a `%%?BeginFontListQuery`, `%%?EndFontListQuery:(*)` pair, and the last item is indicated with a `*`.

8.5.1.2 PostScript Macros

A second configuration file used by *lwsrv* describes the mapping of QuickDraw commands to PostScript commands. Similar to the process with fonts, these macros are downloaded to the printer if they are not in the printer prior to printing a document. The macro files are contained in the directory */usr/lib/adicts*. Each file contains what Apple calls an AppleDict. The AppleDict is really known as a Procedure Set (ProcSet) under the PostScript document structure conventions.

8.5.2 Enabling lwsrv

To set up the Mach^{Ten} spooling system to act as a LaserWriter emulator for remote Macintosh applications, perform the following steps:

1. Configure your printer following instructions in sections "8.2 Local Printing" or "8.3 Remote Printing" (previously described).

NOTE: The */etc/printcap* entry used by *lwsrv* must have an explicit printer designation using either an *rm*, *at* or explicit local printer series of specifications. Without this configuration, a logical print loop will result which has *lwsrv* handing its output to *lpd* and *lpd* handing its output to *lwsrv*. Explicit declaration of a printer for *lpd* is important in avoiding this recursive configuration. Naturally, this is true of multiple *lwsrv*, *lpd*, *lpd*...lpd, lwsrv situations as well, and should be carefully understood when picking a name for lwsrv to advertise to its AppleTalk zone.

When a Mach^{Ten} system is using a remote *lwsrv* to support local *lpd* operations, the local */etc/printcap* must have an *at* entry and *rl* entry in the form:

```
:at=mylaser\:LaserWriter@*:rl:
```

indicating the remote LaserWriter is a "synthetic system" and unable to support normal maintenance PostScript requests such as PostScript page count. Refer to section "8.4 Selecting an Alternate Printer".

2. If the printer you are accessing is a LaserWriter or LaserWriter Plus, proceed to the next step. Otherwise, while still on Max (per the example), obtain the font list from the printer and install it in a file in */usr/lib/<something>Font*. Use one of the techniques described in the section "8.5.1.1 Font Lists".

3. Start *lwsrv*. The command:

```
/usr/lib/lwsrv\
-n mylaser \
-a /usr/lib/adicts \
-f /usr/lib/LWPlusFonts \
-l /usr/adm/lwsrv-errs -p lp
```

   is used to start a local LaserWriter spooler which advertises its name as *mylaser*, with ProcSet directory */usr/lib/adicts*, with font set */usr/lib/LWPlusFonts*, error output file */usr/adm/lwsrv-errs* and local *lpd* printer *lp*. (If you want *lwsrv* to be invoked every time you start Mach^{Ten}, this command should be placed in your */etc/rc* file.) Once invoked, this command causes other Macintoshes on the local AppleTalk segment to "see" a new LaserWriter named *mylaser* for selection with their Choosers. Any subsequent Macintosh application print operations will be sent to the *mylaser lwsrv* which will spool it to its local *lpd* printer (named *lp*) for printing.

4. Go to a Macintosh that does not have *lwsrv* running and open the Chooser. Verify the entry for the spooled printer *mylaser* in the Chooser LaserWriter menu.

8.6 Print Job Status Mail Notification

If there is an access violation or print filter error, any Mach^{Ten} *lpd* will automatically generate electronic mail to the user and host initiating the print request. This feature can be inactivated for any printer by removing the `:MF:` entry from the printer's specification in the */etc/printcap* file. A user may also explicitly specify that electronic mail be sent on successful printing of their print job by adding the *-m* flag to their originating *lpr* request.

8.7 Status and lp Management Programs

The *lpq(1)* program shows the status of all files from the local system queued for printing, as well as the status of the remote printer. Use *lpq* to determine the status of each of your print jobs. Each print job has a number. That number is used to manipulate the job with the *lprm* and *lpc* commands. The example below shows a file queued for printing.

```
lp is ready and printing
Rank     Owner      Job  Files              Total Size
active   holmgren   685  standard input     24799 bytes
```

The *lprm(1*) command is used to delete already queued print files that you have subsequently decided not to print. *Lprm* will first look locally for queued files and then contact any remote spooling systems to request that files be deleted. Files are specified by the job numbers obtained from *lpq*. For example:

```
lprm 685
```

Finally, the *lpc(8)* program is used to perform larger management operations with printers and queues of print jobs. *Lpc* will enable or disable the overall operations of a specific printer, enable or disable the spooling queue of a printer, rearrange the order of jobs in the spooling queue, and find the status of printers, associated spool queues and spooling daemon software.

9.0 Networking with Mach^{Ten}

Mach^{Ten} is rich in features that promote the sharing of data and processor resources with remote systems. Mach^{Ten} communications software supports the exchange of data, remote timesharing of resources, and global electronic mail with a wide variety of systems. Figure 24 shows a Mach^{Ten} system with three Mach^{Ten} windows open. The user has an *rlogin* session in one window and an *ftp* (File Transfer Protocol) in another, and has just sent a message in a third window. Since Mach^{Ten} implements the Internet family of protocols, including the Transmission Control Protocol (TCP) and the Internet Protocol (IP), it is able to interact with other TCP-based microcomputers, workstations (such as a Sun Workstation) and larger scale mini- and maxi-class computers (such as a DEC VAX). The TCP stack in Mach^{Ten} is more robust, better performing, and more fully-featured than Apple’s MacTCP. When Mach^{Ten} is running, Macintosh applications that are accustomed to using MacTCP will automatically use the TCP protocols in Mach^{Ten}. Table 4 shows the differences between Tenon’s TCP stack and Apple’s MacTCP.

Mach^{Ten} internetworking is designed to be self configurable upon installation. However, the installation process assumes that you are installing Mach^{Ten} on an existing internet, that you already have an internet address, and that you know the internet address of your gateway and your domain name server if they exist.

Consult the following “setting up” sections for steps which need to be performed to set up or change your network environment and to keep your network running: “9.1 Using Mach^{Ten} as an Internet Host”, “9.2 Using Mach^{Ten} as a Domain Name Server”, and “9.4 Using Mach^{Ten} as a Router”.

Table 4: Mach^{Ten} TCP/IP Protocol Stack (Comparison with MacTCP)

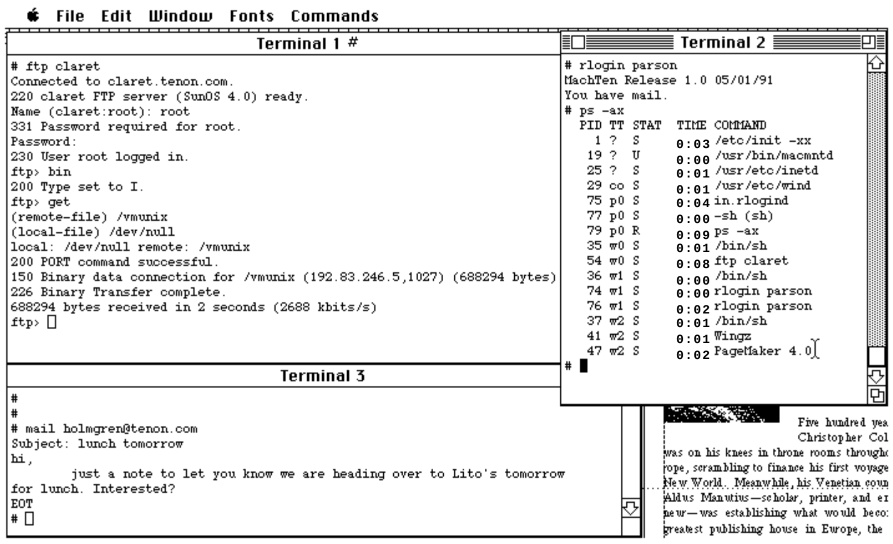
	Mach^{Ten}’s TCP	**MacTCP**
Performance	Buffering strategies, window management, and packet retransmission algorithms are designed to maximize both LAN and WAN performance.	Caters to LANs.
Multihoming	Supports multihoming. Supporting multiple network interfaces lets you have redundant network connections and lets Mach^{Ten} act as a gateway to forward IP packets.	Not supported.
Maximum # of TCP Connections	No limitation.	Limited to 64 (with 16 reserved).

	Mach^{Ten}’s TCP	**MacTCP**
Control Over Interface Options	Provides a range of management options, such as ARP table management, statistics reporting, and reconfiguration without rebooting.	No intrinsic control. Many individual shareware packages are available to fulfill various control functions.
Routing Control	Allows many routing entries and supports dynamic routing configuration via *routed* and *gated*.	Supports a single default route.
Domain Name Resolution	Supports multiple domain name servers (local and remote) and uses a standard ‘hosts’ file.	Supports single (remote) domain name server and uses a non-standard ‘hosts’ file.
Simultaneous Support of UNIX and Macintosh Applications	Yes.	No.
Multicast	Yes.	No.
Loopback Operation	Yes.	No.
IP Aliasing	Supports multiple IP addresses on a single network interface.	No.

9.1 Using Mach^{Ten} as an Internet Host

Mach^{Ten} supports internet communications. The Internet is a collection of networks that provides global communications. Each host on an internet is assigned an internet address which consists of a network identifier and a host identifier. Mach^{Ten} supports the full range of internet class A through C internet addresses, as well as internet subnetting capabilities. To obtain an internet address for your network you must contact Network Solutions in Herndon, Virginia (*http://www.internic.net*). It is a good idea to get an "official" internet address, even if you don't plan to connect to the Internet. This will ease the transition downstream, if and when you decide to join the global network.

Once your system is set up as an internet host you will be able to use the networking protocols (Telnet, SMTP, FTP, HTTP, etc.) to communicate with other hosts that are accessible over your internet. Even if you are not connected to the global Internet you can benefit by using these protocols to form an internal company-wide internet.



```
# ftp claret
Connected to claret.tenon.com.
220 claret FTP server (SunOS 4.0) ready.
Name (claret:root): root
331 Password required for root.
Password:
230 User root logged in.
ftp> bin
200 Type set to I.
ftp> get
(remote-file) /vmunix
(local-file) /dev/null
local: /dev/null remote: /vmunix
200 PORT command successful.
150 Binary data connection for /vmunix (192.83.246.5,1027) (688294 bytes)
226 Binary Transfer complete.
688294 bytes received in 2 seconds (2688 kbits/s)
ftp>
```

```
# rlogin parson
MachTen Release 1.0 05/01/91
You have mail.
# ps -ax
  PID TT STAT  TIME COMMAND
    1 ?  S     0:03 /etc/init -xx
   19 ?  U     0:00 /usr/bin/macmntd
   25 ?  S     0:01 /usr/etc/inetd
   29 co S     0:01 /usr/etc/wind
   75 p0 S     0:04 in.rlogind
   77 p0 S     0:00 -sh (sh)
   79 p0 R     0:09 ps -ax
   35 w0 S     0:01 /bin/sh
   54 w0 S     0:08 ftp claret
   36 w1 S     0:00 /bin/sh
   74 w1 S     0:00 rlogin parson
   76 w1 S     0:02 rlogin parson
   37 w2 S     0:01 /bin/sh
   41 w2 S     0:01 Wingz
   47 w2 S     0:02 PageMaker 4.0
#
```

```
#
#
# mail holmgren@tenon.com
Subject: lunch tomorrow
hi,
        just a note to let you know we are heading over to Lito's tomorrow
for lunch. Interested?
EOT
#
```

Five hundred yea
Christopher Col
was on his knees in throne rooms through
rope, scrambling to finance his first voyage
New World. Meanwhile, his Venetian coun
Aldus Manutius—scholar, printer, and e
new—was establishing what would beco
greatest publishing house in Europe, the

Figure 24. Resource Sharing: FTP, mail, rlogin

In Mach^{Ten} your host name and internet address are from information supplied in the Mach^{Ten} Control Panel. The translation of your host name to an internet address is found in the file */etc/hosts*. This file is set up automatically when the system boots.

Mach^{Ten} employs two methods of translating host names to addresses — by using the */etc/hosts* table and by using a domain name resolver to query a centralized, automated domain name server. You can disable a translation method or specify which method is used first in the */etc/net_search_rules* file. (See *net_search_rules(5)*). Refer to section "9.2 Using Mach^{Ten} as a Domain Name Server" for an explanation of the domain name system. If your internet does not use domain name service, add an entry for each host on your network in */etc/hosts*.

9.1.1 The Mach^{Ten} Networking Control Panel

Mach^{Ten} allows for a multi-homed internet networking environment — permitting multiple independent networks to be connected to a single system. Mach^{Ten} supports several physical interface types, including Ethernet, SLIP, PPP, TokenRing and LocalTalk, in addition to a logical local loopback interface. The Mach^{Ten} Networking Control Panel is a convenient way to configure an interface on AppleTalk, Ethernet and TokenRing for internet networking. Refer to section "3.0 The Mach^{Ten} Control Panel" for information on configuring the primary interface for each network attached to your system. If you want to set up a PPP or SLIP interface, refer to sections "9.13.8 Using PPP" and "9.13.9 Using SLIP".

9.1.2 Configuring Multiple Interfaces

Mach^{Ten} supports multiple interfaces of the same type, including up to four Ethernets, four TokenRings, and multiple SLIP and PPP interfaces. If you have multiple interfaces on Ethernet or TokenRing, you will need to configure those interfaces in */etc/rc* using *ifconfig(8)*. Mach^{Ten} permits up to four Ethernet and four TokenRing interfaces, named `ie0-ie3` and `tr0-tr3`, respectively. For example, the command:

```
ifconfig ie1 joe up -trailers
```

configures the second Ethernet interface installed in your system. The host name *joe* should be added to your */etc/hosts* file along with a unique internet address assigned to the second Ethernet interface.

9.2 Using Mach^{Ten} as a Domain Name Server

Mach^{Ten} is designed to use the Internet Name Server protocol to translate host names into internet addresses. The Mach^{Ten} domain name service (DNS) invokes the Berkeley Internet Name Domain (BIND) software (via *named(8),* the name daemon) to match host names to internet addresses. When you install Mach^{Ten}, the Mach^{Ten} installer, in conjunction with the Mach^{Ten} Control Panel, automatically sets up a table of host names and internet addresses in the file */etc/hosts.* If all the appropriate information was available when you installed Mach^{Ten} and you are using a Domain Name Server, the host table should be ready to go; otherwise you may have to hand-tailor this file as outlined in the following section. The "Name Server Operations Guide for BIND" section (included in the System Manager's Manual section on the Mach^{Ten} CD) tells you how to set up a subdomain and how the name server works.

For a system using a Domain Name Server, the host table only needs the host name and internet address of your Domain Name Server and, if you are connected to the internet via an IP gateway, the name and internet address of the IP gateway.

If your environment does not support a name server, you must make sure that your */etc/hosts* table has all of the host names and corresponding Internet addresses that you will need.

9.2.1 Configuring Mach^{Ten} as a Domain Name Server

If you want to configure one of your Mach^{Ten} system to be a name server, you need to compose data files that describe hosts in your domain, and define domain name service parameters. The Mach^{Ten} distribution contains example name server data files in the */etc/named* directory. If you are configuring the network software manually, you may need to import a full host table from another established host on the network. Over time, this file may need to be edited to add or subtract hosts. See Figure 23 for an example */etc/hosts* file.

```
# Example host file
#
127.0.0.1 localhost
#
# Class C address is subnetted to allow 4 subnets of 62 hosts each;
# subnet mask 255.255.255.192
#   subnet 0 is the ethernet subnet
#   subnet 1 is the appletalk subnet
#   subnet 2 is the slip subnet
#   subnet 3 is reserved
#
# Suffix convention for naming the address of a host interface:
#   _a for AppleTalk, _e for Ethernet, _s for SLIP
# The hostname sans suffix identifies the preferred (or only) address.
#
# The following machines have both Ethernet and AppleTalk interfaces
#
223.255.254.1     mumm         mumm.some.com     mumm_e
223.255.254.65                                   mumm_a
#
# The remaining machines are AppleTalk only
#
223.255.254.96     classic     classic.some.com     classic_a
223.255.254.97     jazz        jazz.some.com        jazz_a
223.255.254.98     rock        rock.some.com        rock_a
#
# Root domain name servers
#
10.0.0.51      SRI-NIC.ARPA
26.0.0.73      SRI-NIC.ARPA
128.102.16.10  NS.NASA.GOV
26.3.0.103     A.ISI.EDU
128.20.1.2     BRL-AOS.ARPA
192.5.25.82    BRL-AOS.ARPA
192.5.22.82    BRL-AOS.ARPA
26.1.0.13      GUNTER-ADAM.ARPA
128.213.5.17   C.NYSER.NET
10.1.0.17      TERP.UMD.EDU
```

Figure 25. Example */etc/hosts*

It is convenient to keep your active files in a subdirectory of */etc* ; for example, */etc/nsdata*. If you are interconnecting to a public or regional network, you must also follow their domain registration procedures.

The name server, *named(8)*, reads a boot file at start up. This file tells the server what kind of name server it is, what domains (zones) it has authority over, and where to get its initial data. The name */etc/named.boot* is assumed unless named is initiated with a *-b* option, such as *-b /etc/nsdata/boot*. The example boot file in */etc/named* follows:

```
; Boot file for yourzone primary master name server
;
directory       /etc/nsdata
primary         yourzone                    zone
primary         255.254.253.in-addr.arpa  zone.rev
primary         127.in-addr.arpa          local
cache             .                         cache
```

To enable named at startup, set the START_named variable in */etc/rc.conf*.

9.2.2 The Domain Name Resolver

An associated component of the domain name system, the domain name resolver, allows the */etc/hosts* table to be reduced simply to the host name and internet address of your name server and your gateway. Each time you attempt to send electronic mail or open a connection to another host, the resolver queries the name server and furnishes the correct address. The file */etc/resolv.conf* contains your domain name and the internet address (not the internet name) of your network name server:

```
domain your.domain.name
nameserver 253.254.255.101
```

When you install Mach^{Ten}, the only requirement in order for your Macintosh to make use of the resolver is to specify your domain name and the internet address of your name server in the Mach^{Ten} Control Panel. The name server may be a *named* running on your local machine or on another network host. Mach^{Ten} will automatically create the */etc/resolv.conf* file when launched. If you want the resolver to also query a backup name server, you simply edit the */etc/resolv.conf* file and add the name server address in the form shown above.

9.3 Using Mach^{Ten} as a Web Server

The NCSA hypertext transfer protocol daemon (*httpd*) serves World-Wide Web (WWW) documents from your Mach^{Ten} system. The *httpd* daemon is pre-configured for standalone operation with configuration files in */etc/httpd*. To turn your Macintosh into an internet WWW server, simply type the following from the *root* account:

```
httpd
```

To automatically start *httpd* during Mach^{Ten} boot up, edit the */etc/rc.conf* file, changing the line `START_httpd` to `START_httpd=”yes”`, and restart Mach^{Ten}.

For detailed documentation on NCSA *httpd*, as well as general information on creating Web documents, refer to the online links in your default home page. Your home page is available for viewing by your favorite Web browser (such as Mosaic or Netscape™) at URL *http://localhost* and for editing in the local file */etc/httpd/htdocs/MyHomePage.html*.

The BBEdit text edit utility contains some useful extensions for expanding upon your default home page and for creating other documents in HyperText Markup Language (*HTML*) format for your Web server. Refer to the BBEdit documents located in the *Utilities* folder on the Mach^{Ten} CD-ROM for more information.

9.3.1 Importing Macintosh Image Files

Images (such as GIF or JPEG files) created by Macintosh applications for use with *httpd* must have a "MUMM / BINA" file creator / type. Use the *chcreat(1)* and *chtype(1)* utilities to change an image file's creator and type:

```
chcreat MUMM <image.gif>
chtype BINA <image.gif>
```

9.3.2 Multihoming Your Mach^{Ten} Web Server

The Mach^{Ten} NCSA *httpd* supports multiple, simultaneous independent Web sites on a single Macintosh. This is useful if, for example, you operate HTTP servers on behalf of several independent organizations. Each IP address on your Macintosh is assigned a domain name associated with the organization on whose behalf you run the Web server. Mach^{Ten} achieves this unique level of service in two ways:

First, Mach^{Ten} allows you to run multiple instances of *httpd* running simultaneously on a single multi-homed (more than one network interface) host, with each *httpd* instance "bound" to a particular network interface and IP address in your system. Each *httpd* server is entirely independent, and has a different set of configuration files.

Second, a Macintosh host that is physically single-homed (one network interface) can be configured to appear multi-homed. Up to twenty unique "virtual" IP addresses and domain names may be aliased to the single physical interface. This allows you to run one instance of the *httpd* server, with the Web server's home page and document directory changing depending on the local IP address alias received on a particular incoming HTTP connection request.

For more information on configuring a multi-homed Web server under Mach^{Ten}, refer to the README file in the */etc/httpd* directory and the sample configuration file in:

```
/etc/httpd/conf/httpd-multihome-sample.conf
```

9.4 Using Mach^{Ten} as a Router

The Mach^{Ten} internet routing service is typically used to connect Mach^{Ten} systems on an AppleTalk network to Mach^{Ten} systems and other UNIX systems on an Ethernet. By supplying an Ethernet interface to any Macintosh running Mach^{Ten}, that Macintosh can serve as a router between a set of Ethernets and a set of AppleTalk networks. The Mach^{Ten} system providing the routing service must be connected to both the AppleTalk network and the Ethernet. The other hosts (Mach^{Ten}, UNIX, and others) need to know which Mach^{Ten} host is the routing server so that they can send packets destined for other networks to the routing server. The routing server then forwards those packets on to their destination network. The command used to tell Mach^{Ten} the address of the routing server is the *route(8)* command. Mach^{Ten} systems on either network will have full connectivity to each other and to other systems supporting Internet protocols, NFS, and remote printer access.

Consider the example in the section “8.5 Extending Remote Printing to Macintosh Applications” of an AppleTalk network and an Ethernet each supporting Mach^{Ten} hosts, as well as other hosts. All Mach^{Ten} hosts on the AppleTalk network must have an internet address with the same network identifier and a unique host identifier. Correspondingly, each host on the Ethernet must have an Internet address with the same network identifier — which is different than the network identifier of the AppleTalk network — and a unique host identifier. The routing server must have two internet addresses, one for its connection on each of the two networks. For the example below, assume that the router’s address on AppleTalk is `A.1` and that the router’s address on Ethernet is `E.4`.

Each internet host (other than the Mach^{Ten} routing server) needs to add a route to its attached network using the *route* command. This command could be executed at any time, but typically it is put in the */etc/rc* file and is executed each time Mach^{Ten} is booted. Each host on the Ethernet needs to invoke the *route* command as follows:

```
route add net  A.0   E.4   5
```

where, `A.0` means all addresses on AppleTalk.

The *route* command tells the Mach^{Ten} kernel to route all packets destined for the AppleTalk network to the Ethernet address `E.4`. The final parameter is a heuristic used by the router to evaluate the desirability of a route. Although it is not needed by the router in this trivial routing case, it is necessary to include it as a parameter to the *route* command.
Each AppleTalk host (other than the routing server) needs to add a route to the Ethernet network using the following command:

```
route add net  E.0  A.1  5
```

where, `E.0` means all addresses on Ethernet.

This command tells the Mach^{Ten} kernel to route all of the packets destined for the Ethernet to AppleTalk address `A.1`.

Non-routing Mach^{Ten} systems refer to routers as *gateways*. Configure your system with a default gateway in the Networking screen of the Mach^{Ten} Control Panel. Other routes may be added using the *route(1)* command.

9.4.1 Invoking IP Forwarding

If a host connected to more than one network desires to forward (route) IP datagrams, check the Enable IP Forwarding box in the Mach^{Ten} Control Panel. If the setting of the check box is changed, the new setting takes effect after the system is rebooted.

9.5 Using MacTCP Applications With Mach^{Ten}

The Mach^{Ten} *mactcpd* library supplants the MacTCP driver during Mach^{Ten} operation. Subsequent MacTCP application requests are captured and directed to run over the Mach^{Ten} protocol stack. This allows Macintosh application programs designed to use MacTCP to run with Mach^{Ten} by using the TCP/IP protocol stack under Mach^{Ten}. Users may launch MacTCP programs while simultaneously running standard UNIX Internet applications (such as *rlogin*) available to users of Mach^{Ten}.

The *Mactcpd* library is launched automatically by Mach^{Ten} during Mach^{Ten} startup.

9.6 Using Mach^{Ten} as a POP Mail Server

A Post Office Protocol (POP3) server is pre-configured in */etc/inetd.conf*. To enable this service, check the "Enable Incoming Connections" box in the Mach^{Ten} Networking Control Panel and restart Mach^{Ten}.

NOTE: The POP Server operates in conjunction with *sendmail*, the electronic mail server described in the next section.

9.7 Using Electronic Mail

Mach^{Ten} provides a complete electronic mail service using user and server Simple Mail Transfer Protocol (SMTP). SMTP is one of the most popular and most widely-used electronic mail protocols in the world. Users on a single Mach^{Ten} system are able to send electronic mail messages to one another. In addition, users logged into a Mach^{Ten} system can send electronic mail to any other system that is both "network accessible" and supports SMTP. There are thousands of networks and tens of thousands of UNIX and non-UNIX hosts that support SMTP.

Electronic mail service in Mach^{Ten} is provided by a number of application and daemon programs. Users interact with front-end applications such as *mail(1)* to compose, review and file messages. Routing, aliasing and forwarding activities are handled by *sendmail(8)* and its delivery agents.

During the Mach^{Ten} installation process, *sendmail* is configured to support direct mail activities on the local Macintosh. Subsequent changes may be made to accommodate other strategies. (See the section entitled "Configuring and Managing Sendmail" in the System Manager's Manual section of the Mach^{Ten} CD-ROM.

Mail(1) is a flexible interactive application for composing, reading, and filing electronic messages. During composition, *mail* allows editing and reviewing of outbound messages, and the inclusion of text from files and / or received mail. While reading messages, *mail* provides commands to browse, display, save, delete, and respond to messages. *Mail* has numerous options to tailor its services to your preferences. These options are set in the *.mailrc* file and are stored in your home directory. An example *.mailrc* file follows:

```
# MachTen standard mail customization file.  When named .mailrc and
# found in your home directory, Mail will use these settings.
#
# run long messages thru more, 20 lines a pop
set crt=20
# tell mail to show 20 lines of headers a pop
set screen=20
# ask for subject line for each message
set ask
# always ask for cc recipients at end of each msg
set askcc
# when sending to an alias list, include self
set metoo
# set mail archive folder name
set folder=MailFolder
# set shell type for any ! or ~!
set SHELL=/bin/csh
# set editor name for ~e command
set EDITOR=/usr/ucb/vi
# an example personal mailing list
# mail to goodfriends will be delivered to all in list
alias goodfriends joe sally bill
```

If you want to experiment with *mail*, try typing:

```
%mail <your_login_name>
```

The *mail* program will respond with:

```
Subject:
```

and await your input. Enter a subject, followed by a carriage return, and then type a short message to yourself. End the message with a period on a blank line. For example:

```
Subject:  Test
This is a test message.
.
```

The message will be sent to you. You can see the message simply by typing:

```
    % mail
```

with no user name. A list of the messages to you will be printed out. The list will indicate who sent the message, the date it was sent, the size of the message, and a few words from the subject heading. The list is formatted as follows:

```
% mail
Mail version 5.5 6/1/90.   Type ? for help.
"/usr/spool/mail/laura":   3 messages 3 new 3 unread
U 1 jdineen@NNSC.NSF.NET   Thu May  2 13:56 1025/16318 "Introduction to.. "
U 2 laura                  Fri May  3 10:05  14/346    "Re: Lunch"
```

You can read each message simply by typing the message number or a carriage return. After reading a message you can delete it by typing `d`, followed by a carriage return. Or, you can save the message to a file by typing `s` followed by a file name. The *mail* program can be exited by typing `x` or `q`.

The "Mail Reference Manual" in the User's Supplementary Documents section on the Mach^{Ten} CD-ROM will help you use all of the features of the Berkeley UNIX *sendmail* that is included with Mach^{Ten}. In addition, there are public domain mailers, such as Eudora[†], that work very well with Mach^{Ten}.

9.8 Using FTP

Mach^{Ten} supports both client and server File Transfer Protocol (FTP) to support the exchange of data on a file-by-file basis with a wide variety of UNIX and non-UNIX computers. Mach^{Ten}-based Macintosh clusters, as well as other processors, can exchange binary and text files using FTP. The FTP file exchange is controlled by user name and password access. Given the proper authorization, remote sites can transfer files to and retrieve files from your Macintosh, and you, in turn, can transfer and retrieve files from any connected system that supports FTP. See *ftp(1)*.

Mach^{Ten} will interoperate with Macintosh FTP applications such as *Fetch*.[††] Binary transfers between Mach^{Ten} and Macintosh FTP applications should be performed in AppleSingle mode, the Macintosh foreign file format used by Mach^{Ten}. Refer to the section "6.11 Accessing Macintosh Files from UNIX Applications" for more information.

[†] *ftp://ftp.qualcomm.com/quest/eudora/mac*
[††] *ftp://ftp.dartmouth.edu/mac*

9.9 Using Mach^{Ten} as an Anonymous FTP Server

A pre-configured, secure anonymous FTP directory hierarchy is in the */home/ftp* folder. The default configuration of FTP permits anonymous FTP access to the */home/ftp* directory. If you would like to disable anonymous FTP service, modify your password databaseto exclude the "ftp" line, or uncheck the "Enable Incoming Connections" box in the Mach^{Ten} Networking Control Panel, and retstart Mach^{Ten}. If you leave the anonymous FTP service in place, documents for public consumption should be placed in the */home/ftp/pub* folder.

9.10 Using Telnet

Mach^{Ten} supports both user and server Telnet, enabling each Mach^{Ten} system to access remote systems and to offer timesharing services to others. Telnet allows you to access other systems from your Macintosh via terminal emulation. This terminal-style capability lets you access processor and memory-intensive UNIX applications on a centralized Macintosh, eliminating the need to upgrade each Macintosh with memory, disk, and other resources. In addition, Mach^{Ten} supports terminal access to non-Macintosh UNIX systems (and non-UNIX systems supporting Telnet) for accessing applications and data on those machines. See *telnet(1)* .

9.11 Using a Distributed File System (NFS)

Mach^{Ten} implements client and server Network File System (NFS). NFS allows you to create a network-wide file system. Files on one system can be accessed on a block-by-block basis by remote systems. Server NFS allows you to configure a Mach^{Ten} system as a centralized file server for a cluster of Macintoshes or other NFS-aware machines.

For detailed information about NFS and setting up clients and servers, see section "7.1 Using NFS" .

9.12 Using Berkeley r-series Commands

Mach^{Ten} supports the Berkeley *r-series* communication utilities. The *r-series* utilities provide a simple way for Berkeley UNIX systems to communicate. The premise is that users in an environment with multiple Berkeley UNIX systems may want a simple, low-overhead way to share programs and files. Each Berkeley system has an */etc/hosts.equiv* file which contains a list of hosts that have permission to remotely log in to your host. This provides a way for others that you trust, or that may have accounts on your system, to log in across the network without having to face a password challenge each time they need to access your machine.

If you have an account "steve" on hostA and an account "steve" on hostB, you can simply type:

```
rlogin hostB
```

and hostB will recognize you as "steve". You can read your mail, compile a program, or do anything you have permission to do on hostB and, when you are finished, simply type:

```
exit
```

Likewise,

```
rcp hostB:<filename>   <filename>
```

will copy a file from hostB to your hostA. You can look up each command (*rlogin(1)*, *rcp(1)*, *rsh(1)*) using the online man pages.

These Berkeley protocols are more streamlined than the Internet protocols, but bear in mind that the Internet protocols work across a wide range of heterogeneous systems.

9.13 Using Serial Line Communications

There are two serial ports on the back of a Macintosh. These ports can be connected to ASCII terminals, modems, or other Mach^{Ten} and UNIX systems. Mach^{Ten} supports traditional UNIX use of these serial ports and provides a comprehensive set of applications for communications via these ports.[†]

9.13.1 Cabling and Connectors

The serial ports require a Macintosh-specific connector: a mini 8-pin connector. Macintosh dealers carry a modem cable which adapts a mini 8-pin connector to the traditional RS-232 DB25 connector which can be directly connected to most modems. Typically a null modem is also needed to cross the send and receive lines between a Macintosh and an ASCII terminal or other UNIX systems.

9.13.2 Matching Serial Ports to UNIX Devices

Mach^{Ten} has two devices for accessing serial ports: */dev/ttya* (used to communicate with the modem port) and */dev/ttyb* (used to communicate with the printer port). Two corresponding devices, */dev/ttyfa* and */dev/ttyfb* are used when your application requires hardware flow control on the modem or printer port. Note that the printer port is usually used as the AppleTalk networking port. If LocalTalk is being used, */dev/ttyb* cannot be used. In order to use */dev/ttyb*, AppleTalk must be turned off (via the Chooser) and the Macintosh must be rebooted.

[†] Mach^{Ten} supports additional serial ports provided by add-on serial port multiplexer boards. Refer to the *Serial Ports* technical note in the */pub/tech_notes* directory on *ftp.tenon.com* for more information.

9.13.3 Baud Rates Supported

UNIX systems traditionally support baud rates from 50 to 9600 baud. The Macintosh hardware does not support baud rates below 300 baud, but it does support baud rates higher than 9600 baud[†]. To take advantage of these higher baud rates, Mach^{Ten} translates requests for the traditional UNIX baud rates to the more useful set of baud rates supported by the Macintosh. Table 5 lists the actual baud rate used by Mach^{Ten} when the traditional baud rates are requested.

Table 5: Baud Rates Supported

Requested Baud Rate	**Traditional UNIX Name <*devtty.h*>**	**Mach^{Ten} Baud Rate**
0	B0	9600 (default)
50	B50	14400
75	B75	28800
110	B110	57600
134	B134	115200
150	B150	230400
200	B200	300
300	B300	300
600	B600	600
1200	B1200	1200
1800	B1800	1800
2400	B2400	2400
4800	B4800	4800
9600	B9600	9600
19200	EXTA	19200
38400	EXTB	38400

[†] If Serial DMA is version 2.0.1 or later. (Serial DMA is included on the Mach^{Ten} CD-ROM.)

9.13.4 General Purpose Interface (GPI) Serial Cable

Mach^{Ten} also is able to support the Macintosh General Purpose Interface (GPI) serial input. With properly constructed cabling (see Figure 26), Mach^{Ten} will be able to simultaneoulsy support hardware flow control and carrier detect modem control transitions.

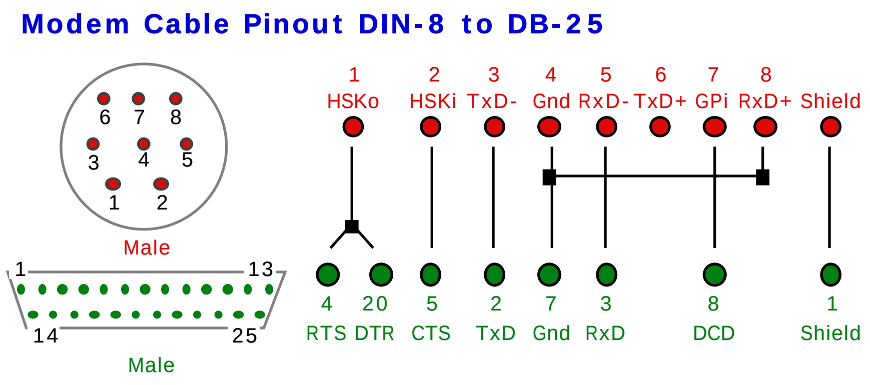


Figure 26. General Purpose Interface (GPI) Serial Cable

9.13.5 Connecting a Modem to a Mach^{Ten} Serial Port

For incoming calls to a modem connected to the serial port, the governing Mach^{Ten} software is *getty(1)*. Since *getty* looks for ASCII text strings, a modem that uses ASCII commands, such as a Hayes compatible modem, will normally require some initialization to keep the responses and status messages from the modem from being handed to *getty* as a login string. A sample Hayes string[†] used for inbound UUCP connections is `1M1L3e0q1`. This string, either typed at the terminal console by the user, or sent to the modem by the software in play, instructs the modem to answer on the first ring, not to echo characters, and not to send result codes. If you depend on the Mach^{Ten} end of a connection to go onhook first at the end of a call, a Hayes compatible modem must be configured to heed the DTR (Data Terminal Ready) line. (The Hayes default is to ignore DTR.)

Outbound calls will normally be governed by a particular UNIX program, such as *uucp(1)*, *tip(1)*, or *cu(1)*, or a Macintosh environment communications program. These programs normally take responsibility for configuring the modem and no special configuration action by the user is required beyond following the instructions for the particular program.

[†] This string is in */usr/lib/uucp/dstr1* on your system. See the documentation accompanying your modem for modem-specific features and commands.

9.13.6 Connecting a Terminal

ASCII terminals can be connected to the serial ports on the back of a Macintosh. These terminals may be used as time-shared login ports. Note that ASCII terminals cannot be used to execute Macintosh programs, but can execute any UNIX program provided by Mach^{Ten}.

9.13.7 Configuring the System to Support an ASCII Terminal

After connecting an ASCII terminal with the proper cabling to the selected serial port, Mach^{Ten} must be properly configured to communicate with the terminal. The */etc/ttys* and */etc/gettytab* files are the database for connected terminals. The simplest configuring case requires editing the */etc/ttys* file for the selected port. Change the string "off" to "on" and select the appropriate baud rate in the "std.XXXX" string. This is all the configuring necessary for most "dumb" ASCII terminals. For the more difficult ASCII terminals, see *ttys(5)*, *termcap(5)*, *gettytab(5)*, and *getty(8)*. In order to tell *init(8)* about the changes to the terminal database, execute the command `kill -HUP 1`.

Once the port is configured and the terminal connected, hit a few carriage returns on the connected terminal. */etc/getty* is listening on that port and looking for a login string and password. Hitting carriage returns will result in a login prompt appearing on the screen. Log out via *logout* or ^D depending upon what shell you are using.

9.13.8 Using PPP

The Point-to-Point Protocol (PPP) was developed to support serial communications in a multiprotocol environment. SLIP, as its name implies, is specifically for IP communications. PPP, on the other hand, as described in RFC 1331, "The Point-to-Point Protocol for the Transmission of Multi-protocol Datagram Links", is designed to encapsulate protocols other than just TCP/IP. Because of its enhanced features, including password authentication, PPP is rapidly becoming the standard for serial communications by Internet service providers.

9.13.8.1 Setting up Mach^{Ten} PPP on Your Macintosh

This section describes how to configure your Mach^{Ten} system to listen for PPP connections on your Macintosh modem port (as a PPP Server) and to connect to remote PPP servers (as a PPP Client).

9.13.8.2 Configuring Your Mach^{Ten} System as a PPP Server

The first step in setting up a PPP server is to connect one or more modems to your Macintosh. The recommended procedure is to use the Communications ToolBox to handle the modem configuration. Test each modem connection by using the *tip* command to dial out:

```
        tip cm.out
```

To exit the *tip* program, press the <return>-<~><.> keys, and then press <return> again. If you wish to access the serial ports directly, the Macintosh “modem” port is “ttya” and the “printer” port is “ttyb”. The “tip” commands are:

```
        tip ttya
```
or
```
        tip ttyb
```

Mach^{Ten} supports hardware flow control for the devices “ttyfa” and “ttyfb”. Note that hardware flow control requires a special cable.

If you cannot dial out, no one can dial in. Once you have established that your modem is indeed connected and accessible, it is time to configure for dial in access. The file */etc/ttys* controls which lines are used for dialin by the *getty* program. The first few lines already have entries. You will have to add more entries if you have more lines. Change the status flag from “off” to “on” to enable a line. Whenever you make changes to the ttys file, you must also signal the *init* program to act upon those changes. A “HUP” signal to the *init* process — process #1 — will accomplish this. Use the following command to signal the *init* process:

```
        kill -HUP 1
```

If you are using the Communications ToolBox, you must first dial in using the “cm.in” line and save its parameters to create the “cm.in1” line.

- Edit */etc/ttys* and set the status for “cm.in"“to “on”.
- Signal the *init* process.

  A Communications ToolBox dialog should appear.
- Adjust the settings for your modem.

  You should then get a “Waiting” message.
- Dial in to your modem.

  You should get a “Login:” prompt.
- **IMPORTANT!** While you have the “Login:” prompt on the machine you are dialing in with, return to Mach^{Ten} pull-down the File Menu.

- Select the last item in the menu.

  You will be presented with a standard "Save" dialog box.

Save *cm.in1* into your *dev* folder. The file name must be seven characters or less. Use *cm.in2*, *cm.in3*, etc., for more lines.

- Edit */etc/ttys* to break the dial in connection.
- Set the status for "cm.in" to "off", and set the status for "cm.in1" to "on".
- Signal the *init* process.

  At this point you should have consistant dialin access to your Mach^{Ten}.

The next step is to setup login accounts for your PPP clients. The user *ppp* is set by default. If you want clients to have their own passwords or internet addresses, use the *adduser* command to create more Mach^{Ten} users. The key is to use */etc/pppserver* as the "shell" for PPP client accounts. The following command will setup user *ppp1*:

```
adduser ppp1 -s /etc/pppserver
```

Once dial in access has been established and the PPP accounts have been set up, it is time to edit the */etc/pppserver* script. You will probably want to print out this script and study it a bit before making any changes. It contains complete instructions. In a nutshell, you must set your Mach^{Ten} address to the "SERVER" variable and your client's address to the "CLIENT" variable. There is a debugging flag which is initially set to "DEBUG_ON". The *pppd* program will send many messages to your console when a client connects. This can be disabled by setting the "$DEBUG_OFF" to the "DEBUG" variable.

The easiest way to choose client addresses is to use addresses from you local subnet. The *proxyarp* option is used by */etc/pppserver* to make this work.

To terminate a PPP connection from Mach^{Ten}, first determine the process number for the *pppd* program by typing the following command:

```
ps -ax
```

Terminate the *pppd* process by typing the following:

```
kill <pid>
```

where <*pid*> is the process number of the *pppd* process.

9.13.8.3 Configuring Your Mach^{Ten} System as a PPP Client

The first step in setting up a PPP client is to connect a modem to your Macintosh. The recommended procedure is to use the Communications ToolBox to handle the modem configuration. Test the modem connection by using the *tip* command to dial out:

```
tip cm.out
```

If you wish to access the serial ports directly, the Macintosh "modem" port is "ttya" and the "printer" port is "ttyb". The *tip* commands are:

```
tip ttya
```

or

```
tip ttyb
```

Mach^{Ten} supports hardware flow control for the devices "ttyfa" and "ttyfb". Note that hardware flow control requires a special cable.

If you are using the Communications ToolBox, using the "cm.out" line will cause an Communications ToolBox dialog to appear.

- Adjust the settings for your modem and dial out.

  You should then see your remote system's prompt.

- **IMPORTANT!** While you have your remote system's prompt, pull down the Mach^{Ten} File Menu and select the last item in the menu.

  You will be presented with a standard "Save" dialog.

Save *cm.out1* into your *dev* folder. Use *cm.out2*, *cm.out3*, etc. for dial-outs to other phone numbers. To exit the *tip* program, press the <return>-<~><.> keys, and then press <return> again.

The file */etc/remote* controls which lines are accessable to the *tip* command. The first few lines already have entries. You will have to add more entries if you have more dialouts. The */etc/pppclient* script does not use */etc/remote*.

Once dial out access has been established, it is time to edit the */etc/pppclient* script. You will probabley want to print out this script and study it a bit before making any changes. It contains complete instructions.
In a nutshell, you must set your dialout line to the "LINE" variable and your login account and password to the "CHAT" variable. There is a debugging flag which is initially set to "DEBUG_ON". The *pppd* program will send many messages to your console when a client connects. This can be disabled by setting the "$DEBUG_OFF" to the "DEBUG" variable.

You should use the *tip* command to manually login to your PPP server so that you can take exact notes on the interaction before trying to compose the *chat* script.

To terminate a PPP connection from Mach^{Ten}, first determine the process number for the *pppd* program by typing the following command:

```
ps -ax
```

Terminate the *pppd* process by typing the following:

```
kill <pid>
```

where <*pid*> is the process number of the *pppd* process.

Running the */etc/pppclient* script when a PPP connection is already in place will return status information about the connection.

Note that */etc/pppclient* can be run from */etc/rc.local* to start up a connection at boot time. It can also be run from *crontab* to ensure that connections are re-established if the line drops.

The default configuration is for the PPP server to supply the client with a dynamic address. There is a “CLIENT” variable in */etc/pppclient* which can be set to a static address for the client.

9.13.8.4 IP Addressing Strategy for Multiple PPP Clients

The client `ppp0` interface may be configured with the same IP address as the `ie0` interface on the client. If the client has no interfaces other than the `ppp0` interface, its IP address should be assigned on a different subnet than the IP address of the PPP server.

The IP addresses for the client and server in our example are specifically chosen so that the PPP client’s IP address is on a different subnet than the PPP server’s IP address. With a netmask of `0xffffff00` on its ethernet (`ie0`) interface, the server is on the `128.0.2` subnet and the PPP client is on the `128.0.3` subnet. Point-to-Point networks do not require the specification of a netmask.

Additional clients which also use the server’s dial-in PPP line should also have IP addresses on the same subnet as this PPP client. This configuration permits a single static route (on the server’s gateway) to handle all of the routing for the dial-in PPP clients on the server’s network (see below).

Run the script per the instructions in the example above to establish the PPP connection.

- On the server’s network, add a static route to the client subnet in your default gateway machine’s */etc/rc.local* file:

```
route add net 128.0.3 128.0.2.1 1
```

All other machines on the server’s network should have default routes pointing at the gateway machine, and all PPP-connected client machines should be on the `128.0.3` subnet.

9.13.9 Using SLIP

Mach^{Ten} supports internet connections over full duplex serial lines using SLIP, the Serial Line Internet Protocol. Mach^{Ten} can be configured to initiate connections to a remote SLIP server system. Mach^{Ten} can also be configured to accept connections from hosts requesting SLIP service. Mach^{Ten}'s SLIP capabilities combined with Macintosh hardware and any of a number of commercially available modems make an excellent platform for accessing large dialup networks such as PSINet and UUNET. Other Internet Access Providers provide SLIP services as well. Mach^{Ten} SLIP is equally effective as a simple extension of your office network to your home over residential telephone lines.

The information in this chapter provides information on configuring, using, and maintaining SLIP connections under Mach^{Ten}.

9.13.9.1 Configuring Mach^{Ten} To Dial-Out To SLIP Servers

Mach^{Ten} provides a program called *tip(1)* to automatically establish a full-duplex connection to another machine via a dial-out modem attached to a serial port on your Macintosh. Once connected, *tip* can automatically initiate a login authentication exchange with the remote system and, if successful, enable a SLIP session on the line. If the remote system requires any manual intervention to start up SLIP (as with PSINet networks), refer to section "9.13.10.5 Manual Connections to SLIP Servers".

During the SLIP session, the SLIP line or "interface" becomes an integral part of the Mach^{Ten} networking environment, running stand alone or simultaneously with other network interfaces such as Ethernet or Appletalk configured into your Macintosh. Both Mach^{Ten} and Macintosh Internet applications have full access to the SLIP network interface.

To enable SLIP dial-out network connections, the command:

```
tip -s <system name>
```

is used, where `<system name>` is a label for a configuration database entry describing exactly how to contact and communicate with the remote system. The *tip* configuration database is described below.

9.13.9.2 Tip Activity Log

The file */usr/adm/aculog* is an administrative file which logs all dialing activity done by *tip*. It can be used to track who is calling which numbers and for how long. Mach^{Ten}'s default installation does not create */usr/adm/aculog* and no logging will be recorded if this file does not exist. Execute the following commands to create this file (as *root*):

```
touch /usr/adm/aculog
chmod 666 /usr/adm/aculog
```

9.13.9.3 The Tip Configuration Database

The */etc/remote* file (see *remote(1)*) is the primary database file used by *tip* to describe SLIP hosts and the modems set up used to gain host access. Mach^{Ten} is shipped with a SLIP entry for a hypothetical system named *joeslip* in */etc/remote* which you will need to modify depending on your system configuration:

```
joeslip:\
:dv=/dev/ttya:br#2400:cu=/dev/ttya:at=hayes:pn=5551212:du:\
:sa=89.0.0.2:sm=225.0.0.0:da=89.0.0.1:st=slip\
:ls=/etc/login.script.joe:
```

Each field in the entry is delimited by colons and describes the following characteristics:

joeslip	The system name assigned to the connection. This is used when *tip* is invoked to establish the connection *tip -s joeslip*
br#	The serial port speed (baud rate) designator for the SLIP connection. Modify this field as necessary to match the baud rate of your modem according to the following table:

br# entry	**Modem Baud Rate**
50	14.4 Kbps
75	28.8 Kbps
110	57.6 Kbps
300	300 bps
600	600 bps
1200	1200 bps
1800	1800 bps
2400	2400 bps
4800	4800 bps
9600	9600 bps
19200	19.2 Kbps
38400	38.4 Kbps

Note that some of the very low baud rates have been mapped to the recently more popular higher baud rates.:

dv	The serial port with an attached or built-in modem on your Macintosh to be used for the SLIP connection. The standard modem port (*/dev/ttya*) is the default setting. You may also use the printer port (*/dev/ttyb*); however, tthe printer port is typically reserved for AppleTalk connections.
at	A string representing the modem type.
pn	The telephone number dialed to establish the connection. If no phone number is given, *tip* will use the system name to look up a phone number in the */etc/phones* database. (See *phones(5)*).
sa	The local SLIP internet address.
da	The destination SLIP internet address.
sm	The SLIP network mask.
st	The protocol type. By default, SLIP.
ls	A pathname to a special script that is run by *tip* after the connection is established. The script typically communicates account name and password information to the remote host’s login shell to authenticate the pending SLIP session. Refer to section “9.13.9.5 Tip Login Script” for details on the script’s command format.

9.13.9.4 Hardware Flow Control

You will need to enable hardware flow control on your serial line depending on the capabilities of your modem and your RS232 cable type. Apple makes at least two different types of RS232 cable; one lacks the Clear To Send (CTS) RS232 signal necessary for flow control. To enable flow control, set the dv= field in */etc/remote* SLIP host entry to */dev/ttyfa* or */dev/ttyfb* depending on the serial port you have chosen for your modem. The “f” in the Mach^{Ten} device name indicates hardware flow control on that serial port.

9.13.9.5 Tip Login Script

The *tip* login script is used to communicate initial login authentication information to the remote login shell. It is a collection of commands read by a specialized interpreter within *tip* after making a physical connection to the remote SLIP host. The file */etc/login.script.joe* is provided as an example authentication dialog with a remote UNIX *login(1)* shell. The file */etc/login.script.junc* is an example dialogue with a Cisco terminal server using Tacacs.

The basic format of the available script commands are described below:

The "label" command: "label foo"	
The "goto" command: "goto foo"	This command says branch to label "foo".
The "send" command: "send \dFooBar\r"	The characters are sent to the remote host. The backslashed characters follow the "C" programming language conventions with the following additions: \d = delay 1 sec, \s = space, \x = break
The "recv" command: "recv 10/error login:"	This command means: look for the string login:' but if its not seen in 10 seconds, branch to label "error".
The "done" command: "done"	This command indicates a successful login.
The "fail" command: "fail"	This command indicates an unsuccessful login.

Blank lines and lines beginning with '#' are ignored.

Use the *-v* option to *tip* to display the command exchange with a remote host as a method of debugging special login script files.

9.13.9.6 Header Compression

Mach^{Ten} will enable SLIP header compression for initiating SLIP sessions if an `:hc:` field is added to the desired SLIP entry in */etc/remote*:

```
joeslip:\
:hc:dv=/dev/ttyfa:br#2400:cu=/dev/ttya:at=hayes:pn=5551212:du:\
:sa=89.0.0.2:sm=225.0.0.0:da=89.0.0.1:st=slip:ls=/etc/login.script.joe:
```

Header compression is disabled in the default configuration. If header compression is enabled, a corresponding flag should be set in your SLIP dial-in server database. For Mach^{Ten} systems, the dial-in server configuration is described below.

9.13.9.7 Setting a Default Route

Mach^{Ten} will make the remote host in a SLIP connection the default route if no default route exists on your system and if an `:rt:` parameter is added to the desired SLIP entry in */etc/remote*. Refer to *route(8)*:

```
joeslip:\
:hc:dv=/dev/ttyfa:br#2400:cu=/dev/ttya:at=hayes:pn=5551212:du:\
:sa=89.0.0.2:sm=225.0.0.0:da=89.0.0.1:rt:st=slip:ls=/etc/login.script.joe:
```

All packets bound for networks unknown to your Mach^{Ten} system are directed to the default route.

9.13.9.8 Setting the MTU

The MTU (Maximum Transmission Unit) in a SLIP connection is a hard limit on the output packet size. The default MTU setting is a compromise between efficient serial line bandwidth utilization and good interactive performance. For small settings, relatively more bandwidth is consumed transmitting the IP protocol headers with each data packet. Conversely, since any small interactive packets on a connection must wait for in-progress large packets to finish, large MTU settings adversely affect interactive response time.

The default MTU value of 296 bytes was chosen to insure good interactive performance while minimizing the header cost. Adjustments to the MTU can be made in the Networking Screen of the Mach^{Ten} Control Panel. The MTU setting is applied to all configured SLIP interfaces. The use of header compression (see section “9.13.9.6 Header Compression”) is recommended for all MTU settings where supported by your SLIP server.

9.13.10 Configuring Mach^{Ten} As A Dial-In SLIP Server

Mach^{Ten} can be configured as a dial-in server to enable SLIP connections initiated by remote systems. The *sliplogin(8)* program and database files */etc/ttys*, */etc/passwd*, and */etc/slip.hosts* are the primary user configurable components that allow dial-in SLIP service under Mach^{Ten}.

9.13.10.1 Serial Port Set Up

For dial-in SLIP service, a modem must be connected to a serial port on your Macintosh. The port labeled the modem port (named */dev/ttya* under Mach^{Ten}) is the recommended port. The printer port (*/dev/ttyb*) is also acceptable; however, use of the printer port is usually reserved for AppleTalk.

To make Mach^{Ten} aware of your serial port selection, you must modify the */etc/ttys* (see *ttys(5)*) database file. This file contains information that is used to initialize and control the use of the serial port selected, including port speed, type, flow control, and active status:

```
#
# name  getty                     type        status  comments
#
ttya    "/etc/getty std.9600"     unknown     off     secure
ttyb    "/etc/getty std.9600"     unknown     off     secure
```

By default, the Macintosh ports under Mach^{Ten} are set to 9600 baud and disabled (status "off"). By way of example, an entry describing a 2400 baud modem connected to the modem port and activated would appear as follows:

```
ttya    "/etc/getty std.2400"     unknown     on secure
```

Refer to *gettytab(5)* for details on the various types of line configurations available under Mach^{Ten}. To enable flow control on the serial port, change the *ttya* field to *ttyfa*. The "f" in the Mach^{Ten} device name indicates hardware flow control on that serial port. You must either send a hangup signal (see *kill(8)*) to the *init* process (see *init(8)*) or restart your Mach^{Ten} system to activate any modifications made to */etc/ttys*.

9.13.10.2 Login Authentication

When a remote SLIP host initiates a connection on a configured serial port, Mach^{Ten} will start the *login(1)* program which prompts the remote host for a username and password. *Login* will consult the */etc/passwd* (see *passwd(5)*) database file to authenticate the user. If the user is authentic, *login* will start up a special program to enable SLIP on the connection. The Mach^{Ten} */etc/passwd* file contains a default account with no password for user `Sjoe`:

```
Sjoe::1000:10:Slip line from joe:/var/tmp:/usr/etc/sliplogin
```

Modify this entry as necessary for your SLIP configuration.

9.13.10.3 The Sliplogin Program

Sliplogin(8) is the program named in */etc/passwd* to run after a remote login session is established over a potential SLIP line. *Sliplogin* effectively turns the terminal line into a SLIP link to the remote host. To do this, the program searches the file */etc/slip.hosts* for an entry matching the current login name. If a matching entry is found, the line is converted to SLIP line discipline (that is, 8 bit transparent I/O) and configured using information from the entry. For example, header compression is enabled or disabled for the SLIP connection at this time depending on the "compress" or "autocomp" fields in the entry.

Sliplogin then runs a shell script to initialize the SLIP interface with the appropriate unit number, local and remote IP address, and netmask for the connection according to the parameters in */etc/slip.hosts*. The default initialization script is */etc/slip.login*; however, if particular hosts need special initialization, the file */etc/slip.login.loginname* will be executed instead if it exists. The example */etc/slip.hosts* file from the distribution is shown below:

```
#
#  login    local-addr  remote-addr mask        speed       options
#
#  (options are normal,compress,noicmp,autocomp)
#
gladys      rocks_s     gladys_s    0xff000000  19200       autocomp
psinet      psinet_s    psinet_s    0xffff0000   2400       normal
Shavoc      okeeffe     havoc       0xffffff00   9600       normal
Soxford     okeeffe     oxford-gw   0xffffff00      0       compress
Schez       okeeffe     chez        0xffffff00   9600       compress
Smjk        okeeffe     pissaro     0xffffff00   2400       compress
Smjk2       okeeffe     pissaro     0xffffff00   2400       compress
Sjoe        89.0.0.1    89.0.0.2    0xff000000   2400       normal
```

9.13.10.4 SLIP Session Termination

On session termination, *sliplogin* will automatically remove the SLIP interface from the list of Mach^{Ten} network interfaces. Mach^{Ten} will reset the serial port to the configuration specified in */etc/ttys* in preparation for a new SLIP session.

9.13.10.5 Manual Connections To SLIP Servers

Some SLIP servers may not support the automated functions of tip -s (dialup terminal servers or other non-BSD UNIX systems, for example). In this case, it may be necessary to perform some of the SLIP set-up commands manually. If it is possible to use the -s option to *tip* to configure SLIP then that method is recommended. Otherwise, follow the manual set-up steps below.

- Configure an abbreviated entry in */etc/remote* as follows:

```
sliphost:\
dv=/dev/ttyfa:br#2400   (Example baud rate (br#) setting)
```

- Set up the *tip* activity log described in section "9.13.9.1 Configuring Mach^{Ten} to Dial-Out to SLIP Servers" if you want a log of your session.

- Then invoke *tip* without the *-s* option

```
tip sliphost
```

- Then, dial out to the remote host (this example uses a modem with an AT command set, dialing to a hypothetical phone number preceded by one and an area code):

```
atdt15185551212
```

If the number is busy, *tip* may not notice this right away, but it will timeout waiting for a reply.

Subsequent steps in the manual SLIP connection process achieve the SLIP configuration that would otherwise be done automatically by *tip*.

9.13.10.6 Logging In

How you log in depends on the host to which you are making the connection. With some terminal servers you log in with your own name and password, then give the command *slip*. With BSD systems, a special login account is usually created by the system administrator which runs the *sliplogin* program. In this case, the name of the special login account is used, and SLIP is invoked automatically. In either event, once connected, log in using the appropriate account name and password:

```
Username: xxx
Password:
```

If there are any unusual characters printed during the login process, try responding to the `Username:` prompt with a carriage return. Repeat this until the line settles or the autobaud feature selects the correct baud rate.

9.13.11 Starting SLIP

If you have logged in to a terminal server that supports SLIP, your IP address will be reported back to you as part of the log on. On some systems after login verification is complete, it is necessary to type `slip` and note the address that is reported back. This is the IP address assigned by the server to your end of the serial line connection being established. Break out of *tip*, closing the *tip* session, but not hanging up the modem, by typing ~`<Control-Z>`. At this point the message "`cannot synchronize with hayes modem`" may be printed, and the DTR signal from the Macintosh to the modem will be off. This behavior is not a problem as long as the modem doesn't drop the connection, (that is, it stays Off Hook). In addition, your modem should be configured to ignore the Data Terminal Ready (DTR) signal from the Macintosh. Next reserve the tty port for SLIP using *slattach(8)*:

```
/usr/etc/slattach /dev/ttyfa <baud rate>
```

`<baud rate>` sets the speed of the connection, for example 2400 baud. Configure the SLIP interface using *ifconfig(8).* Note that a netmask is not explicitly specified. *Ifconfig* will default to the proper netmask in typical configurations; however, for SLIP interfaces configured as part of a subnet implementation, a proper subnetmask must be specified. `<src ip address>` represents the address returned from the server when `slip` was typed (or during login). For PSINet connections, `<src address>` and `<dest address>` are identical. For logins that run *sliplogin*, `<dest address>` must match the local address entry in the */etc/slip.hosts* file on the remote system.

```
/usr/etc/ifconfig sl0 <src ip address> <dest ip address> up
```

Add the routing table entry for *route(8).* For PSINet the default `<gateway address>` is usually constructed by changing the last byte of the SLIP internet address assigned by the server to 1. Otherwise, the default gateway address is the `<dest address>` passed as an option to *ifconfig*, above. Contact your SLIP server network administrator (or your PSINet or UUNET representative as applicable), to verify that this is true in your case:

```
/usr/etc/route add net default <gateway address> 1
```

9.13.11.1 Testing Connectivity

Once SLIP is running, the Mach^{Ten} host should have full Internet connectivity. Test access to another host using *ping(1).* If you know a nearby local gateway, *ping* it, otherwise try "nic.ddn.mil". If this is successful, you are fully connected.

9.13.11.2 Stopping SLIP

When the SLIP session is over it is desirable to undo the configuring done to bring SLIP up, especially if the Mach^{Ten} host is connected to another network and sessions on this network are going to be started. Also, it could save some money if long distance charges apply.

First, delete the routing table entry:

```
/usr/etc/route delete net default <gateway address>
```

Second, bring down the SLIP interface:

```
/usr/etc/ifconfig sl0 down
```

Third, kill the *slattach* process:

```
      kill -9 `ps -ax | egrep slattach | egrep -v egrep | awk '{print $1}'`
```

And lastly, hang up the phone connection by powering off the modem.

9.13.11.3 C-Shell Scripts to Expedite SLIP Connections

The following script can be used to bring up SLIP once the *tip* connection has been established and the internet address of this SLIP line is known. This script should be called *slipup* and it takes one parameter, the internet address returned by the server host.

Example C-Shell Script to Set Up a SLIP Connection

```
#!/bin/csh
#Use csh because of 'clever' substitution below
#(in route line).
#
#Connect a MachTen host to a server via slip.  This
#script expects the serial connection to have been
#previously established.
#
#$1 is the internet address assigned by the SLIP
#server
#/dev/ttya is the Macintosh modem port.
#The baud rate is set to 2400

#slattach to assign the tty to SLIP

/usr/etc/slattach /dev/ttya 2400&

#Configure the SLIP interface.  Note a netmask is
#not explicitly specified.
#Typically you won't need subnetting on the SLIP
#interface.
#ifconfig will default to the proper netmask in this
#typical configuration.

/usr/etc/ifconfig sl0 $1 $1 up -trailers
#Add the routing table entry.  The default gateway
#address is usually constructed by changing the last
#byte of the SLIP internet address assigned by the
#SLIP server to 1.  Call your network rep or consult
#a local expert to verify that this is true in your
#case.

/usr/etc/route add net default ${1:r}.1 1
```

The following script can be used to bring down SLIP at the end of the SLIP session. This script should be called *slipdown* and it takes one parameter, the internet address returned by the SLIP server.

Example C-Shell Script to Bring Down a SLIP Connection

```
#!/bin/csh
#Use csh because of 'clever' substitution below (in
#route line).
#
#Disconnect a MachTen host from a SLIP server.  Undo
#the work of slipup.
#
#$1 is the internet address assigned by SLIP server

#Delete the routing table entry.

/usr/etc/route delete net default ${1:r}.1

#Bring down the SLIP interface.

/usr/etc/ifconfig sl0 down

#Kill the slattach process

kill -9 `ps -ax | egrep slattach | egrep -v egrep | awk '{print $1}'`
#Don't forget to hang up the phone connection by
#powering off the modem.
echo “Don’t forget to hang up the phone connection by powering off the modem.”
```

9.13.12 Using Communications ToolBox Devices

Mach^{Ten}’s CommToolBox devices provide a UNIX device interface to the Connection Manager component of the Macintosh Communications ToolBox. CommToolBox devices allow UNIX programs to use any Connection Manager Tool for standard input and output, just as if they are using a typical tty device. However, the Connection Manager Tools completely handle and hide the details of the physical connections from the UNIX programs, allowing the programs to concentrate on the logical stream of data to and from the device, and to be oblivious to the underlying hardware properties like modem types, baud rates, flow control mechanisms, etc. The use of Mach^{Ten}’s CommToolBox devices is described below.

The default CommToolBox devices */dev/cm.in* and */dev/cm.out*, correspond to dial-in and dial-out. You may use the *mknod* utility to create additional CommToolBox devices with customized names. CommToolBox devices use the major number 11. The minor number 0 indicates a CommToolBox device that *listens* on the Connection Tool for incoming connections. This is most useful for dial-in modem lines. The minor number 128 indicates a CommToolBox device that *opens* the Connection Tool. This will actively open the Connection Tool and dial-out, if appropriate. Some Connection Tools do not have the dual modes of dial-in and dial-out (e.g., the Serial Tool). These tools usually require an *open* rather than a *listen*, and thus require use of the minor number 128.

9.13.12.1 Creating CommToolBox Devices

The Mach^{Ten} CommToolBox devices support *unconfigured devices* and *configured devices*. Both */dev/cm.in* and */dev/cm.out* are unconfigured CommToolBox devices. An unconfigured device has no stored information indicating to which tool it connects or the specific attributes of any tool. An unconfigured device can thus be used to open a connection to any Communications ToolBox Connection Tool.

When an unconfigured device is opened by a UNIX application, the application is blocked until the device is configured. To enable the user to configure the device, the Communication Manager displays its configuration dialog box in front of any open Mach^{Ten} windows. This dialog box allows the user to select a particular Connection Tool and to configure it. The UNIX application is allowed to continue when the configuration dialog box is closed, and proceeds to *open* or *listen* on the Connection Tool, depending on the minor number of the unconfigured device. If the user cancels the configuration dialog, or the Connection Tool fails to open properly, the device is closed and an error is returned to the UNIX application.

For each successfully configured unconfigured device, a new item is added to the end of Mach^{Ten}'s File menu. These menu items may be selected, which allows that particular configuration to be saved in a file. Files created in this manner become *configured CommToolBox devices*.

Configured CommToolBox devices can be stored anywhere on the Macintosh File System. The typical repository is */dev*, which is where all other devices reside. When a configured device is opened by a UNIX application, the saved configuration is used to create a CommToolBox connection of the same type of Connection Tool as the configuration that was previously saved. This connection is immediately *opened* or *listened* on, appropriately. An error is returned if the Connection Tool specified by the configured device cannot be found or configured. There is no new item added to the end of Mach^{Ten}'s File menu for successfully opened configured devices. The only way to change the settings for a configured device is to open an unconfigured device, configure it appropriately, and save it as a configured device. Configured devices can be discarded when they are no longer needed by dragging them to the Trash.

By default, Mach^{Ten} supports up to three simultaneously opened (or listening on) CommToolBox devices.[†]

9.13.12.2 Configuring UNIX Communications Programs For The CommToolBox Devices

UNIX serial communications programs like *tip*, *pppd*, and SLIP usually require intimate knowledge of the physical connection media. The attributes of each connection are often stored in a collection of configuration files. The classifications of these attributes are built into each program, and often lag behind the rapidly-changing characteristics of today's communication mechanisms. Since other *logical* connection attributes are also stored in these UNIX configuration files, it is not yet possible to completely remove the need for these databases. However, Mach^{Ten}'s CommToolBox devices allow most UNIX communications programs to be configured with only a minimal set of logical attributes, and still work with a variety of physical connection media. This simplification can greatly reduce the effort required to configure and maintain communications with new advanced modems or other physical connections.

[†] Additional CommToolBox devices may be configured by modifying a resource in the Mach^{Ten} application. Refer to the *Configuration Resources* technical note in the */pub/tech_notes* directory on *ftp.tenon.com*.

Each of the following examples shows a minimal logical configuration for the most common UNIX serial communications programs. These examples demonstrate the use of unconfigured CommToolBox devices which enable the minimal configuration to be used to open any Connection Tool. This way, the desired Connection Tool itself can be used to select the specific attributes of the physical connection. Once the Connection Tool is operating properly, its configuration may be saved in a configured CommToolBox device and this device can be used in place of the unconfigured device shown in these examples.

Using PPP with CommToolBox Devices. *Pppd* does not use any device configuration database, but rather takes all of its arguments on the command line. A minimal *pppd* command line to passively listen for PPP connections on any Connection Tool can be invoked as follows:

```
# /usr/etc/pppd -p -d /dev/cm.in 101.0.0.2:
```

A minimal *pppd* command line to actively open a PPP connection on any Connection Tool can be invoked as follows:

```
# /usr/etc/pppd /dev/cm.out 101.0.0.1:
```

Using Tip with CommToolBox Devices. *Tip* is designed to timeout on hard-wired connections that do not open in less than 10 seconds. Since Connection Tools abstract the hardware details from the UNIX programs, *tip* believes CommToolBox devices are hard-wired. To prevent *tip* from timing out on CommToolBox device opens, the attribute `:ct#0:` should be added to the */etc/remote* entry for the CommToolBox device. This attribute has been added to the *tip* program specifically for use with the Mach^{Ten} CommToolBox devices. It is not described in the *tip* manual page. The value 0 indicates no timeout is necessary. Any other value indicates a timeout in seconds. The default used if no `:ct:` attribute is specified is 10 seconds.

A minimal */etc/remote* configuration to open a *tip* session on any Connection Tool follows:

```
cm.out:dv=/dev/cm.out:ct#0
```

To open a *tip* session using this configuration, invoke the following command:

```
# tip cm.out
```

Using Getty with CommToolBox Devices. A minimal */etc/ttys* entry to enable *getty* to listen on any Connection Tool for incoming login sessions follows. In this case, the 9600 baud rate specified here will be ignored by the Connection Tool. The baud rate specified in the Connection Manager's Configuration dialog box will be the actual baud rate used.

```
cm.in    "/etc/getty std.9600"  vt100     on secure
```

Getty is automatically started by *init* and need not be invoked from the command line. If you make a change to the */etc/ttys* database, you can inform *init* of this change using the following command:

```
# kill -HUP 1
```

Using SLIP with CommToolBox Devices. The easiest way to establish a SLIP connection is via the *-s* option provided in *tip*. A minimal */etc/remote* configuration to open a SLIP connection on any Connection Tool follows:

```
joe.slip:\
    :dv=/dev/cm.out:hc:ct#0:\
    :sa=89.0.0.2:sm=225.0.0.0:da=89.0.0.1:st=slip:\
    :ls=/etc/login.script.joe:
```

To establish a SLIP connection using this configuration, invoke the following command:

```
# tip -s joe.slip
```

9.13.12.3 CommToolBox Notes

Hardware Flow Control. It is necessary to have the proper cable for your type of modem and the corresponding Connection Tool in order for hardware flow control to function properly. There are several different modem cables manufactured for the Macintosh, and they are definitely not equal with respect to flow control signals. (See picture earlier in this document.) Hardware flow control and detection of dropped Carrier Detect (CD) are supported. If you require both hardware flow control and detection of dropped CD, use the CommToolBox devices.

Dial-In Dial-Out. Mach^{Ten} CommToolBox devices do not support simultaneous dial-in and dial-out on the same Connection Tool. This capability may be added to a subsequent release of Mach^{Ten}.

10.0 Mach^{Ten} Programming Environment

Mach^{Ten} provides a rich UNIX software development environment. The Mach^{Ten} software tool suite has been modified to work in real memory for Macintosh hardware. The traditional UNIX software development tools — *ld*, *nm*, *ar*, etc. — have been ported to run in PowerPC native mode and produce PowerPC native mode Code Fragments.

10.1 Software Development Tools

Mach^{Ten} software development tools include:

TOOLS FOR MANAGING SOFTWARE DEVELOPMENT	
Revision Control System (RCS)	Automates the storing, retrieval, logging, identification and merging of multiple revisions of text
TOOLS FOR SUPPORTING SOURCE CODE DEVELOPMENT	
indent	Formats C Code
lex	Generates lexical analysis programs
yacc	"yet another compiler-compiler"
TOOLS FOR BUILDING LIBRARIES AND PROGRAMS	
gas	PPC assembler
cc	GNU CC compiler director: forks the appropriate tools for compiling and loading the objects specified
ld	Combines several object files into one, resolving external references and searching libraries
make	Executes a scripted set of commands to update one or more target files
ranlib	Converts archives to a form which *ld* can load more rapidly
cc1	GNU C language compiler
cpp	GNU preprocessor
gdb	Symbolic debugger
nm	Prints lists of symbols in object files

Libraries	
libc.a	the C library
libcurses.a	Screen functions with cursor motion
libg++.a	C++ library
libm.a	Math library
librpc.a	Remote procedure call library
TOOLS FOR BUILDING X WINDOWS PROGRAMS	
imake	X platform independent *make* environment
libX.a	Core X protocol library
libXt.a	X Intrinsics toolkit library
libXaw.a	The Athena widget set of buttons, pull-down menus, labels, etc.
libXmu.a	X miscellaneous utilities library
libXau.a	X security authorization library
libXext.a	X extensions library
libXi.a	Alternative input device extension library
libXm.a	Motif widget library
libXdmcp.a	X Display Manager Control Protocol library

10.2 Allocating Memory in Mach^{Ten}

Mach^{Ten} allocates memory for its constituent applications in two ways. When an application is loaded into the system for the first time, a Macintosh system module called the Code Fragment Manager (CFM) is used to load and manage application Code Fragments. The CFM handles allocation of memory for new fragments, relocation of internal software references within a fragment and dynamic linkage between different but related Code Fragments.

The CFM uses two areas of memory to store Code Fragments — Mach^{Ten} application memory, known as the Mach^{Ten} heap, and system memory. Mach^{Ten} application data fragments are allocated from Mach^{Ten} application memory. Mach^{Ten} application code fragments are allocated from system memory. In addition, if virtual memory has been enabled, access to pages of code are managed on a demand paged basis so that only software that has been referenced is actually loaded into memory. This facility is called Memory Mapped files.

Once an application is under execution, the applications may make subsequent requests for memory. These requests are allocated from Macintosh system memory.

A dynamic bar chart of allocated system memory, as well as Mach^{Ten} and other application memory, can be displayed by double-clicking the "About This Macintosh" entry in the Finder Apple menu.

Sometimes, due to heavy system loads, either Mach^{Ten} memory or system memory allocations will be completely exhausted. By reallocating the size of the Mach^{Ten} memory space, it is possible to change the relative sizes of the two allocations. See section "10.2.1 Setting the Mach^{Ten} Heap Size" for directions on how to increase the Mach^{Ten} memory heap.

Mach^{Ten} typically fails to allocate memory in two ways — the CFM fails to find sufficient memory when loading an application, or the application itself fails during a secondary memory request. Both failures print an error message. When the CFM fails to find enough memory to load an application, the message is "CFM error -2810". Getting this error means that the Mach^{Ten} memory space is fully allocated; one solution is to quit Mach^{Ten}, increase the Mach^{Ten} heap size and re-launch Mach^{Ten}.

When a secondary memory request fails, an error code is returned to the application and the response to a user is application specific. When a secondary request fails, the system memory space is fully allocated. The basic response here is to quit any Macintosh applications that might also be using system memory, close any unused Finder windows, or ultimately to quit Mach^{Ten} and this time reduce its heap allocation size. Note that by reducing the size of the Mach^{Ten} heap, the size of the allocatable system memory is increased, so it is possible to iterate several times around increasing or decreasing the Mach^{Ten} heap size to best suit a particular system's memory profile.

10.2.1 Setting the Mach^{Ten} Heap Size

Setting the size of the Mach^{Ten} heap varies the sizes of the code and data allocation pools. Increasing the size of the Mach^{Ten} heap generally increases the size of the data allocation memory area at the expense of decreasing the code allocation size. To set the Mach^{Ten} heap size, make sure that Mach^{Ten} is not running. Next find the Mach^{Ten} application icon and highlight it with a single mouse click. Then pull down and double-click the "Get Info" selection in the Finder File menu. If the "Get Info" selection is not selectable, the Mach^{Ten} icon has not be highlighted with a single click. Re-select the Mach^{Ten} icon and the "Get Info" selection should become selectable. This will display information about the Mach^{Ten} application. The lower right-hand corner of the display will contain a Suggested Size setting for Mach^{Ten}. Select the box and raise or lower the setting acording to your systems needs. Close the display by clicking the close box. Restart Mach^{Ten} and verify the new size by opening the dynamic memory display by double clicking the "About this Macintosh" in the Finder Apple Menu.

10.3 PEF and XCOFF

Mach^{Ten} uses PEF (PowerPC Executable Format) as its binary executable standard. PEF is an Apple Computer standard for defining executable programs and executable code objects. The PEF standard is native to the PowerPC Macintosh architecture. By generating PEF, the Mach^{Ten} development tools achieve a large degree of synergy with other Power Macintosh development environments and the Power Macintosh OS. PEF executable files are composed of one or more elements called Code Fragments. The many advantages of Code Fragments include the runtime dynamic linking between fragments and automatic code and data sharing between applications with shared libraries of Code Fragments.
PEF formated files are developed as the last step in the Mach^{Ten} software production process. Internally the Mach^{Ten} development tools rely on an IBM binary executable standard called XCOFF (eXtended Common Object File Format). This file format is much richer in the amount of information kept about an application and its structure and data at the expense of significantly larger disk file and longer dynamic relocation overheads. For Mach^{Ten} debugging purposes, it is possible to retain an XCOFF image of a PEF executable. See section "10.11 Loader" for command line information to retain XCOFF file images.

10.4 Shared Libraries

The Mach^{Ten} development software supports the production of libraries of software that may be shared among application programs. Using shared libraries, a second copy of a program shares part of its memory with a first program instance. Shared libraries result in large memory savings when families of applications are executing at the same time. Secondarily, shared libraries also aid in the introduction of a new version. Since shared libraries are linked during the execution of a program rather than the development of a program, a new version of a shared library can be introduced simply by replacing an older version and restarting the applications that use it. In older non-shared organizations, each application would have to be re-linked with the new library and then introduced into a system.

10.4.1 Shared Library Production

To produce a shared library with the Mach^{Ten} development system, you must specify that you want a shared library to be produced by using the *-Xlsharedlibrary* flag on the compile or loader command line. You must also specify which symbols within the library that you want exported for linkage with an application. The command line flag *-Xlexpall* exports all global symbols in the shared library. The command line flag *-Xlexp=<filename>* exports the symbols named in the file *<filename>*. For example, if the file *some_symbols* contained the following lines:

```
symbA
symbB
symbC
```

using the command line flag *Xlexp=some_symbols*, the three symbols would be found in the shared library and would be set up for export and linkage when an application program requested that the library be included in its import file list.

10.4.2 Run-Time and Compile-Time Libraries

The specification of the *-Xlsharedlibrary* flag will create two libraries — a run-time library and a compile-time library. The run-time library should be put in the run-time search path of the loader that is launching the application which requested that library for shared library import. In the case of Mach^{Ten}, which uses the Mac OS Code Fragment loader, the directory */MachTen/MachTen Libraries* has been explicitly added to the search path for the MacOS Code Fragment loader, hence the run-time version of the Mach^{Ten} shared libraries should be put in this folder. The compile-time library should be used on the compile command line of applications during software compilation. The run-time library serves the purpose of providing a single copy of software to multiple applications. The compile-time library is used to resolve compile-time references between imported symbols in an application and exported symbols from a shared library.

10.4.3 Run-Time and Compile-Time Naming Conventions

It is important to understand the naming convention for the run-time and compile-time libraries. Compile-time libraries are named by adding a *.a* suffix to the output file specified on the command line. For example, the command line:

```
cc -o shrlib -sharedlibrary -expall *.o
```

will produce a run-time shared library in the file *shrlib*, and will produce a compile-time shared library in the file *shrlib.a* that exports all symbols.

10.5 Traditional UNIX Libraries

The traditional UNIX libraries (*libc.a, libm.a*, etc.) are implemented as shared libraries under Mach^{Ten}. These compile-time libraries reside in */usr/macppc/lib* and the corresponding run-time libraries reside in */MachTen/MachTen Libraries.*

When an application is created, the */usr/macppc/lib* libraries must be used as the definition library to satisfy compiler Header Call definition requirements. When an application is launched, the */MachTen/MachTen Libraries* version of the library will be found by the Code Fragment Manager and loaded into memory for dynamic loading.

Some of the libraries in */usr/macppc/li*b do not have a twin in */MachTen/MachTen Libraries.* These libraries are not implemented as shared libraries, hence the loader will statically link with these libraries and there is no need for a run time version.

If another Power Macintosh software development environment (MPW, Metrowerks, etc.) is used to build Mach^{Ten} applications, the PEF libraries in */MachTen/MachTen Libraries* should be used for both the compile time Header Call definitions and the run time dynamic linking. The XCOFF libraries in */usr/macppc/lib* are usually not acceptable input to these Power Macintosh software development tools.

10.6 File System Hierarchy

Mach^{Ten} consists of a combination of programs from 4.4BSD, GNU and other sources. The sources for each of these programs is contained in */base/src/bsd4.4*, */base/src/gnu*, and */base/src/misc*. The executables for these programs are typically stored in */usr/bin* or */bin*. Tools that depend on a particular target system†, e.g. the Power Macintosh tools, reside in */usr/macppc/bin*.

These directories contain tools that run on the same Mach^{Ten} system (i.e., all of the tools in */usr/macppc/bin* are native mode PPC applications), their placement being subject to the difference being the target system of the tool's output. The tools in */usr/macppc* produce native mode PPC applications. Selection of which tools are run is controlled by environment variables and symbolic links.

The 4.4BSD and GNU environments use different *Makefile* formats (see section "10.7.5 Makefiles"). Mach^{Ten} supports both formats. 4.4BSD and GNU *Makefiles* use a target, or object directory, that is separate from the source directories. For these targets, Mach^{Ten} uses */base/macppc/bsd4.4* and */base/macppc/gnu*, respectively. The path */var/obj* is frequently used in the 4.4BSD *Makefiles*; for the 4.4BSD programs it should be a symbolic link to */base/macppc/bsd4.4*.

† This version of Mach^{Ten} supports development of PPC executables only. Future versions will enable developers to target multiple platforms.

The following figure shows the key components of the Mach^{Ten} file system hierarchy for software development. The */bin* and */usr* paths are the traditional UNIX paths for the executable programs and header files specific to this instance of Mach^{Ten}. The */base* path is the highest level root for the source of a complete distribution of Mach^{Ten} (or other software). This path also provides directories for building the object specific to a particular target platform.

```
/bin
/usr
            /bin
            /include
            /lib
            /macppc/
                        /bin
                        /include
                        /lib
            /mac68k/
                        /bin
                        /include
                        /lib
/base
            /src
                        /bsd4.4
                        /gnu
            /macppc
                        /bsd4.4
                        /gnu
            /mac68k
                        /bsd4.4
                        /gnu
```

When porting or developing new UNIX programs for Mach^{Ten}, you should mimic or extend this file system organization if it doesn't otherwise adversely affect the task of porting your project's source and *Makefile* structure. Following this model may actually ease the job of porting software to the Mach^{Ten} system, especially considering synergy between the provided 4.4BSD master *Makefiles* in */usr/share/mk*, developing for multiple targets, and sharing your work with other Mach^{Ten} users.

In a latter chapter of this document, the source for the UNIX game *rogue* is used as an example of a program that can be compiled, debugged, and run on a Power Macintosh using Mach^{Ten}. This source is in */base/src/bsd4.4/games/rogue*, which further illustrates the structure of the 4.4BSD source file hierarchy used in Mach^{Ten}.

10.7 Cross-Development Tools And Targets

The *macppc* component of the target path names used in the 4.4BSD and GNU *Makefiles* is necessitated by the future use of the Mach^{Ten} cross-development capabilities for either the 68K or Power Macintosh, (i.e., */base/mac68k/bsd4.4* would be the target directory for building the 68K versions of the 4.4BSD applications). The locations of the development tools is also dictated by the cross-development capabilities of Mach^{Ten}.

Note that all of the "machine specific" software development tools reside in */usr/macppc/bin*, the libraries in */usr/macppc/lib*, and header files in */usr/macppc/include*. The non-machine specific pieces are in the traditional */usr/bin*, */usr/lib*, and */usr/include* paths. Links to the machine specific development tools are set up in the corresponding directories.

10.7.1 Pre-Defined Names

The following names are used extensively in the Mach^{Ten} header and source files. These names are pre-defined appropriately in the Mach^{Ten} development environments, and should be used (as appropriate) when developing for either the 68K or PPC release of Mach^{Ten}:

```
__MACHTEN__          /* defined in all MachTen environments */
__MACHTEN_68K__      /* defined when developing for the MachTen 68k env */
__MACHTEN_PPC__      /* defined when developing for the MachTen PPC env */
```

10.7.2 Programming Notes[†]

- The predefined macro `__MACHTEN__` may be used to conditionally include code specific for the Mach^{Ten} environment.

- The predefined macro `__MACHTEN_PPC__` may be used to conditionally include code specific for the Mach^{Ten} PowerPC environment.

10.7.3 Default Mach^{Ten} Environment

Power Mach^{Ten} may be used to develop code for either the 68K or PPC environment. The selection of the target in a cross-development environment is controlled by the `PATH` and other environment variables and symbolic links.

Since Mach^{Ten} provides support for PowerPC targets, this environment is established as the default. More information on specifying the target in a cross-development environment will be provided when the multiple target capability is released.

[†] Additional information on generating programs for Mach^{Ten} is available in a *Porting* technical note in the */pub/tech_notes* directory on ftp.tenon.com.

10.7.4 Header Files

Mach^{Ten} header files are ANSI compliant. Most of the header files are common between the 68K and PPC releases. The machine-specific portions of these files are demarked using the pre-defined constants:

```
#ifdef __MACHTEN_68k__
            /* 68K specific code */
#endif

#ifdef __MACHTEN_PPC__
            /* PPC specific code */
#endif
```

A few header files are completely different for the two environments and are contained in separate directories in the header file hierarchy (either */usr/macppc/include* or */usr/mac68k/include*). Symbolic links are used to select the appropriate header files when developing for either of the Mach^{Ten} environments.

10.7.5 Makefiles

MachTen supports both the 4.4BSD and GNU environment *Makefile* formats, and Mach^{Ten} supports both. The 4.4BSD environment provides a set of master *Makefiles* that can be easily included in a project for integration into a 4.4BSD release. These master *Makefiles* reside in */usr/share/mk*. The path */var/obj* is frequently used in these 4.4BSD master *Makefiles*. In an initial Mach^{Ten} installation, this path is set up to the target directory for applications to be built for the same architecture as the one on which Mach^{Ten} is running (i.e., on Mach^{Ten}for the Power Macintosh it defaults to */base/macppc/bsd4.4*, .) The GNU *Makefiles* are largely self contained, with each project containing its own complete *Makefile* structure.

10.8 GCC

Gcc has been completely integrated into Mach^{Ten} as the default compiler. A symbolic link from *cc* to *gcc* is provided for compatibility with UNIX tradition. *Gcc* may be used to completely handle compilation, loading, and producing XCOFF and PEF outputs. A list of the Mach^{Ten} PowerPC tools that *gcc* calls, either directly or indirectly, is given below.

```
cpp, cc1, gas, ld, mkpef
```

The default behavior for *gcc* on Mach^{Ten} follows UNIX tradition. For example:

```
cc -o hello hello.c
```

produces an executable PEF file in *hello*. The C compiler interprets files with a *.c* suffix as normal C language input. Files with a *.i* suffix are interpreted as previously processed by *cpp* and passed directly to *cc1*.

10.9 C++

G++ is used as the C++ compiler for Mach^{Ten}. G++ is a C++ "front-end" for GCC using the same code generator, assembler and loader back-end combination. G++ uses *cc1plus* instead of *cc1* used by GCC. The C++ compiler uses several suffixes to detect C++ input — *.cc, .cxx, .cpp, .c++* and *.C*. Files with a *.ii* suffix are interpreted as previously processed by *cpp* and are passed directly to *cc1plus*.

10.10 Fortran

Mach^{Ten} includes a Fortran front-end adapted for the PowerPC from the G77 Fortran compiler. Like G++, G77 has a special front-end to handle code syntax and intermediate language generation, but relies on the same back-end tools to complete the development of PEF files. Fortran source files are distinguished by .f, .for or .F file name suffixes.

10.11 Loader

The Mach^{Ten} loader *ld* accepts and produces IBM binary standard format XCOFF files. It relies on an external program called *mkpef* to translate XCOFF output to Apple PEF format for use on the PowerPC. Typically, the XCOFF version of the file is deleted after the PEF file is produced. Optionally, the XCOFF file may be saved by adding the *-Xlxcoff* parameter to the *ld* execution command. The XCOFF output is stored in a file with the same name as the target PEF file, with a trailing *.xcoff* (i.e., *basename.xcoff*). For example, if the loader is producing a file called *myprog*, the XCOFF version of the loader's output will be put in *myprog.xcoff*. This program is automatically called by *gcc* (in the default case), so typically you will not have to execute this program or build specific rules into your Makefile structure.

Since Mach^{Ten} runs in real memory, stack sizing considerations enter into the development of a program. Mach^{Ten} provides a -*Xlstack <stackbytes>* parameter that provides the ability to set aside memory for the application stack. This value has a default of 36KB, which works well for most applications. However, some applications can have stack memory requirements that approach several megabytes and need their stack parameters set accordingly. This capability is also provided by the CFRG resource, which is discussed further in the "Stack Overrun" portion of the section "10.13 Real Memory Issues".

10.11.1 mkpef

The *mkpef* program takes the XCOFF output of the Mach^{Ten} loader and produces a PEF executable image. This program is automatically called by the Mach^{Ten} loader (in the default case), so typically you will not have to execute this program or build *mkpef* rules into your *Makefile* structure.

10.12 Symbol Information

Mach^{Ten} application symbol information is obtained by using the *nm* program. *Nm* works with XCOFF files optionally output from the compilation of an application. Use *-Xlxcoff* on the compilation command line to produce XCOFF output. Files containing XCOFF information are typically named with a *.xcoff* suffix. The command:

```
nm -n afile.xcoff
```

will produce symbol information relative to location zero for application data and text.

10.13 Real Memory Issues

Where possible, every effort has been made in Mach^{Ten} to provide as complete a 4.4BSD UNIX environment as possible. However due to the lack of memory protection in MacOS, Mach^{Ten} is a "real memory" implementation of UNIX. Therefore, it cannot possibly mimic all of the "features" of virtual memory. Most applications are oblivious to the subtle differences between running under virtual memory or real memory. These differences are documented here so the programs that do depend on virtual memory can be easily diagnosed as such, and modified to run on Mach^{Ten}.

Not having virtual memory means that programs deal with physical addresses and have limited available memory and a limited stack. The following real memory problems are known to exist:

sbrk	Some programs that call *sbrk()* make the assumption that each successive call to *sbrk* will return memory addresses contiguous to the previous call. This is not possible to support in a "real memory" environment, thus these programs require modifications to work with Mach^{Ten}. In Mach^{Ten}, *sbrk* translates into a heap-oriented memory allocation call.
page boundary	Typical virtual memory implementations allocate memory on page boundaries, thus requests to allocate less than some minimum amount actually allocate this minimum. Some programs may accidentally depend on this by accessing past the minimum memory length requested, without causing problems. In a real memory implementation, memory is a scarce resource, and memory requests are typically rounded to a much smaller size, like a multiple of 16 or 32 bytes. Accidentally accessing past this real memory boundary will most certainly cause problems to the actual owner of this space.

stack overrun	Mach^{Ten} supplies each process with a fixed size stack. If a subroutine defines lots of storage on the stack (automatic data in C), this could cause stack overrun, which will almost assuredly cause the system to crash. Alternatives for large automatics are to redefine them as static, or to use the *malloc* or *alloca* system calls to explicitly allocate some memory. The default process stack size is 36K bytes. A larger process stack size may be specified using the CFRG resource for the Mach^{Ten} applications. A template CFRG resource is provided in the */usr/lib/temp.pef* file. This template has a stack size of 122K, and a program name *progname*. Use *ResEdit* to copy and paste the CFRG resource contained within this template into the resource fork of your executable image. Use ResEdit to modify the first fragment description's application stack size to your specific requirements and modify the program name to your program name. If your *ResEdit* does not have a template for the CFRG resource, the first fragment description's application stack size is set at offset 48 (decimal) from the beginning of this resource. (Refer to the -*Xlstack* loader option in section "10.11 Loader".)
out of memory during compile	Both the C compiler (*gcc* and *cc1*) and the *make* program (*make*) require a lot of memory to run. If you are compiling large files and have an elaborate program generation environment built up where a *Makefile* causes multiple copies of *make* to run at the same time, this will require large amounts of memory and may cause either the C compiler or *make* to fail during some memory allocation request, aborting the program generation. A potential solution to this problem is to use some of *make*'s more sophisticated features, such as VPATH, and to reorganize the program generation process to be completely controlled from one or two *make* invocations.
fragmentation	In a real memory situation it is possibly to fragment memory in such a way that, although the total free memory is sufficient to satisfy a particular allocation request, none of the individual memory pieces is big enough to satisfy the request. This situation can be aggravated by use of the *realloc()* call, with successively larger sizes.
etext	In a virtual memory environment programs can make assumptions about the relative locations of the programs *text*, *data*, and *bss* sections. In real memory these assumptions simply are not valid. Programs using the symbols *etext*, *edata*, and *end* probably require modifications to not depend on the relative locations of these symbols.

10.13.1 Calculating Memory Requirements

The Code Fragment Loader loads the code (or executable) portion of Mach^{Ten} applications in the MultiFinder heap, and the data portion of the Mach^{Ten} applications in the Mach^{Ten} heap. Therefore, a running Mach^{Ten} application consumes space in both the MultiFinder and Mach^{Ten} heaps, and it is necessary to have space available in both of these heaps in order to successfully launch Mach^{Ten} applications.

The *About This Macintosh* dialog box (under the Apple Menu when Finder is the front application) provides an easy means to approximate how much space is available in both the MultiFinder and the Mach^{Ten} heaps. However, is not easy to predict how much more memory a particular Mach^{Ten} application will require. Mach^{Ten} libraries (*libc*, *libm*, etc.) are shared libraries. Mach^{Ten} applications that reference these libraries share a single instantiation of the library's code. The library's data segments are not shared. Each Mach^{Ten} application allocates a private copy of the data segments for each shared library it references. This dynamic sharing of the Mach^{Ten} libraries, combined with the stack and memory allocations internal to Mach^{Ten} programs, makes it difficult to determine how much additional memory a Mach^{Ten} program will consume from the Mach^{Ten} and MultiFinder heaps.

Examining the *About This Macintosh* display both the before and after launching a Mach^{Ten} application provides a way to approximate memory requirements. Using MacsBug (or another debugger) to dump the exact heap statistics provides a more precise means to calculate memory usage.

10.14 Making Macintosh Applications

One of the benefits of producing PEF formatted binary images is that the Mach^{Ten} software development tools may be used to produce Apple standard application programs. To develop Macintosh applications, the software must use standard Apple interface libraries and Apple standard interface header definition files, it must specify a non-standard startup routine providing for proper startup data definition and proper program exit and follow Apple system programming rules (e.g. WaitNextEvent, MoreMasters etc.).

10.14.1 Macintosh OS Interface Libraries

To include a standard Apple interface library with Mach^{Ten} application development commands requires a few careful preparation steps. The file creator must be modified and, potentially, a translation must be specified from the name of the library file to the fragment name used to access the shared library by the Macintosh system.

System interface libraries are available in all standard Apple software development systems (e.g. MPW, Metrowerks), typically in a libraries hierarchy of directories. Under a libraries hierarchy, there is a set of Power Macintosh interface libraries. To incorporate one of these libraries on a Mach^{Ten} command line, the Macintosh creator must be modified. Generally this has no affect on access to the library by the native development software. Modifying a library creator to "MUMM" will allow Mach^{Ten} to access the file as a binary file. The file should already have a Macintosh type identification of *shlb* which is used by Mach^{Ten}

development tools to determine whether shared access to the PEF interface library is to be reflected in the final application.

File creator and type information is available by running *ResEdit* or *chcreat(1)* and *chtype(1)* and selecting the “Get Information” function from the FILE menu. Creator information can be modified under *ResEdit* with traditional Macintosh point and click methods.

Care must be taken not to make a copy of one of the libraries using Mach^{Ten} applications until the creator has been changed to “MUMM”. Access to a library not created by “MUMM” will automatically create an Apple single version of the copied data and make it unusable by Mach^{Ten} development tools.

Interface library fragment names are the names of the Code Fragments that are used by the Macintosh system to dynamically relate application programs to a shared library. Each shared library has a fragment name which is typically the same as the name of the Macintosh file containing the fragment. A shared library Code Fragment name is contained in the shared library’s CFRG resource. This resource may be viewed and modified with the *ResEdit* application. If the file name and Code Fragment name are the same, then no further action is necessary. The Mach^{Ten} development tools will place a reference in the resulting PEF output to a Code Fragment named with the same name as the name of the Macintosh file containing the Code Fragment.

However, if the library name and Code Fragment name are different, a translation between the two must be specified during the PEF production phase of the Mach^{Ten} development tools. To specify a non-standard file name to Code Fragment name mapping the *mkpef* application must be run as an explicit, separate program step with a mapping specification. To prepare for a separate *mkpef* step, the *-Xlxcoff* parameter must be specified during compilation or during the Mach^{Ten} loading phase. The *-Xlxcoff* parameter preserves the XCOFF formatted output file which will be used as explicit input to the *mkpef* application. The *mkpef* application can then be run with the command:

```
    mkpef -lfilename:=fragmentnamecreator XXXX -type APPL inputfile.xcoff outputfile
```

Multiple file name to fragment name translations can be included on the same command line. A creator of something other than “MUMM” must be specified and a type of “APPL” must also be specified.

10.14.2 Macintosh OS Header Definition Files

Mach^{Ten} Macintosh applications also need access to Macintosh interface header definition files for data/code typing and parameter access. A family of these files is also typically included with every native Macintosh development program under a headers directory hierarchy. Access to these files by Mach^{Ten} development applications requires that these files have a Mach^{Ten} creator of “MUMM” and a “TEXT” type designation. The simplest method of converting a family of header files is to select a whole directory and drag the group of files to the Mach^{Ten} *Unix<->Text* application, drop the group on the application is let it change each of the files. The Mach^{Ten} *Unix<->Text* application is contained in the *utilities* directory of the Mach^{Ten} distribution.

Typically MPW and other development tools do not examine the creator or type of their constituent header or shared library interface files. This means that it is possible to modify the creator and type of one copy of files and have them used by both Mach^{Ten} and traditional Macintosh development applications.

10.14.3 Non-Standard Startup Routine

Standard Mach^{Ten} applications use a startup routine that sets up a UNIX parameter and environment variables, calls a main subroutine and, on return from *main*, call the UNIX standard *exit* system call. A standard Macintosh application has different requirements for startup preparation. The initial routine must declare a QuickDraw variable named *qd*, call the *main* application and, on return, call the Macintosh standard *ExitToShell* system call. A pre-compiled file conforming to Macintosh startup requirements has been included with the Mach^{Ten} distribution under the file name */lib/mac_crt0.o*.

10.14.4 Macintosh Application Construction

The newly modified interface libraries and header definition files must be specified during the compilation and linkage phases of the Mach^{Ten} code production process. This is specified using typical UNIX standard methods. Access to an interface library is specified using the *-L* parameter on the UNIX command line and access to a directory of header definition files is specified using the *-I* parameter on a UNIX command line. The command line:

```
      ld -Iinclude -o hello mac_crt0.o hello.o ../lib/InterfaceLib \
             -Xlcreator MACH -Xltype APPL
```

will create an output file *hello* specifying a search path beginning with the *include* directory. The directory *../lib* will be searched for an interface library named *InterfaceLib*. If *InterfaceLib* is type *shlb* it will be treated as a shared library and accesses to exported symbols from the library will be treated as dynamic linkages to be satisfied during the runtime execution of the application. The *Xlcreator* and *Xltype* parameters specify a “MACH” creator and a standard Macintosh application type. The resulting “point & click” Macintosh program can be executed independently of Mach^{Ten}.

10.15 Debugging

Since Mach^{Ten} for the Power Macintosh uses the Apple-defined PEF format for executable code objects (Code Fragments), there are a number of alternatives for debugging Mach^{Ten} applications on the Power Macintosh. The traditional UNIX approach for Mach^{Ten} will be *gdb*, while Apple and other third party debuggers are also available and will work with Mach^{Ten} PEF Code Fragments.

10.15.1 Debugging Using gdb

The preferred way to debug Mach^{Ten} applications is with the use of the Mach^{Ten} *gdb* application. This debugger provides source and assembly level debugging, complete with data structure display, single step code execution and source code specified break point specification. *Gdb* requires access to both the PEF and XCOFF formats of an executable program. Using the *-g* command during the production of the application will automatically produce PEF and XCOFF files. The command:

```
cc -O -o hello -g hello.c
```

will produce the PEF output file *hello* and the XCOFF output file named *hello.xcoff*. The Mach^{Ten} command:

```
gdb hello
```

will start execution of the *gdb* application which will automatically search for both PEF and XCOFF files. Once both files are found, UNIX standard *gdb* commands may be given to set break points, examine data and complete the debugging process. Further information on the *gdb* application can be found in the *gdb* manual page accessed via the Mach^{Ten} *man* command.

10.15.2 Non-Mach^{Ten} Debugging Tools

First make sure that a debugger is not already installed on your system. Typically, if the message "Debugger Installed" is printed when the system is booting, it means that some form of debugger has already been installed on your system. If you have a debugger, investigate its origins and usage. It is possible that it will provide the debugging support you need.

10.15.2.1 MacsBug

MacsBug† is a Macintosh debugger which offers low-level debugging support on Power Macintoshes. *MacsBug* 6.5d12 works with Mach^{Ten}, providing assembly level debugging, break points and PowerPC exception handling. It interprets traceback data attached to each Mach^{Ten} generated subroutine showing routine name and parameter information.

If you do not already have a debugger installed, it is recommended that you install *MacsBug* 6.5D12 in your *System Folder*. It is located in *Utilities:Apple Utilities f:Debuggers* in the Mach^{Ten} distribution. It will provide you with a backup if one of your applications crashes.

† ftp://ftp.support.apple.com/pub/apple_sw_updates/us/mac/utils

Using *MacsBug* you can display memory as PowerPC assembler code (use a command *ilp@616ab0*). You can set a break point with the command *brp<addr>*. The *g* command is used to resume execution of the application. The *s* and *so* commands are used to single step and single step over (steps over subroutine calls).

By using the CFM command in *MacsBug* when the application is executing, the absolute addresses for both data and text areas are available. The sum of the data base address from the MacsBug CFM command and the offset information from the *nm* command provide an absolute memory location for a variable.

10.15.2.2 The Debugger

The Debugger (commonly referred to as the "Jasik" debugger after its author, Steve Jasik), supports Macintosh style (point & click) source debugging for both 68K and PPC platforms. *The Debugger* consumes approximately 2MB of RAM. Check the folder *The Debugger™* in the *Debuggers* folder of this distribution for more information on obtaining this product.

10.15.2.3 Macintosh Debugger for PowerPC

The *Macintosh Debugger* for PowerPC is Apple's symbolic debugging tool for Power Macintosh. The newest revision (2.0d3) works in a "single headed" mode, which eliminates the requirement of needing two Macintoshes to debug PPC applications. This version of the debugger consumes approximately 5MB of RAM.

Apple's documentation describes some interesting features, including source debugging directly from the XCOFF file (i.e., the debugger has the *makesym* function built in), a flag to enter the debugger each time an application loads a Code Fragment, a method of mapping *xSYM* files to these newly-loaded Code Fragments, and an ability to present and trace Thread Manager stacks (each Mach^{Ten} application uses a Thread Manager Thread).

10.15.2.4 Metrowerks Debugger

The Metrowerks Code Warrior software development environment includes a symbolic source code debugger for Power Macintosh applications. This debugger would be a logical choice for debugging Mach^{Ten} applications if the applications are also being developed using the other Code Warrior tools.

10.16 Programming Example

10.16.1 Rogue

4.4BSD includes a number of games, some of which have a long history in the UNIX world. This example uses the game *rogue* as an example for software development for the Power Macintosh using Mach^{Ten}.

10.16.2 Building the Executable

First it is necessary to build the executable binary image for the *rogue* program. The *Makefile* has already be set up for the Mach^{Ten} environment, so you need only to execute the few steps below. (You should examine the *Makefile* for the details.)

```
cd /base/src/bsd4.4/games/rogue
make
```

The *rogue* executable PEF file is built in *obj/rogue*, and the XCOFF file is stored in *obj/rogue.xcoff*.

10.16.3 Debugging Using MacsBug or Other Macintosh Debuggers

The *rogue* sources have been modified to include a call to enter a Macintosh debugger. This *Debugger()* call is contained within an *#ifdef DEBUG* conditional, so this call is only made if the compile options include the *-DDEBUG* flag (*make COPTS=-DDEBUG*). The *Debugger()* library call has been added to *libc*. When this call is made, control is passed to the debugger and the debugger indicates that the next instruction to be executed is the first line after the *Debugger()* library call.

If you do not have a Power Macintosh debugger installed, do not use the *Debugger()* library call. It would result in a program error (unknown trap), and would cause Mach^{Ten} to quit executing.

10.16.4 Finding the Program

MacsBug provides a *cfm* command to dump the tables internal to the Code Fragment Loader in an easily readable format. Type *cfm* to display the entire table, then find your program and its starting address within the table.

10.16.5 Building a Symbol Table

Use the *nm* program to make a symbol table from *obj/rogue.xcoff*. Redirect the output of the *nm* program to an output file. The *cfm* command presents the text and data starting and ending addresses. Use the text address and the offset from the *nm* output to calculate an absolute address in text space. Use the data address and the output from the *nm* command to calculate an absolute address in data space. You can dump the contexts of a calculated address using the MacsBug *dm <addr>* command.

11.0 The X Window System

The X Window System®, or X, is a network transparent graphics window system originally developed at the Massachusetts Institute of Technology. X is based on a *client/server* model, where an application program (the *client*) communicates information to a display program (the *server*) which then outputs the information to a bitmapped display. The server directs user input, typically from a keyboard or mouse, to the client program for interpretation.

Client programs are commonly run on the same machine as the server; however, clients may also run transparently from other machines connected to the server machine (sometimes called an X terminal) via a network. The client and server communicate with one another using the *X Protocol*, a software communication protocol layered over TCP/IP.

X allows multiple clients to run simultaneously, each displayed in a separate, overlapping *window* on the server. Using the mouse and keyboard, the user controls the size, appearance, and location of each window on the display, or X *desktop*.

The Mach^{Ten} X Window Software supports the X client/server model under Mach^{Ten} on a Macintosh. Using the Mach^{Ten} X Window Software, the Macintosh display is transformed into a high performance X server. Client programs created and run under Mach^{Ten}, as well as network clients, may be displayed on the server. Simultaneous operation of Macintosh applications with X clients is also possible using MultiFinder to switch from the X server to the Macintosh program.

This chapter provides a user with the information necessary to operate and administer the Mach^{Ten} X Window Software. Related documentation on X is listed in section "11.5 References & Suggested Reading".

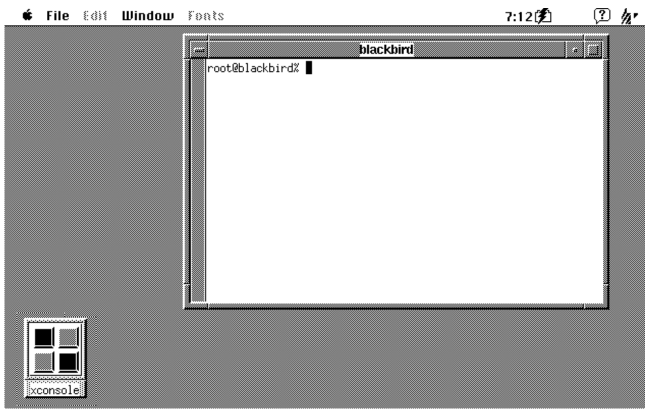
The following section introduces the X environment under Mach^{Ten}. It assumes a completed installation of Mach^{Ten} and the X Window Software software on your Macintosh. The software installation instructions are in section "2.0 Installing Mach^{Ten}" and an overview of the X Window Software components appears in section "11.3 Mach^{Ten} X Window Software Overview".

11.1 The X Desktop

The Mach^Ten X Window Server runs entirely within a window in the Mach^Ten windows environment. For those unfamiliar this environment, refer to Chapter 6.0, "Mach^Ten Windows". When started, the X server will create an X desktop similar in function to the Macintosh desktop on your display. The X desktop will coexist with Mach^Ten terminal windows, other Macintosh applications, and your Macintosh desktop, allowing you to conveniently switch among desktops and applications. X applications that you run and display on your Macintosh will appear on the X desktop, while Mach^Ten terminal and other Macintosh application windows will appear on the Macintosh desktop.

11.1.1 Starting the X Server

To launch the X server and create an X desktop, type the Apple key and the letter *T* simultaneously (<Command-T>). The X desktop appears below:



```
root@blackbird%
```

Figure 27. The X Desktop

The Mach^Ten *wind(8)* process MUST be running in order to start the X server.

The X server desktop window differs from a Mach^Ten terminal window in that the familiar Macintosh title bar, close box, zoom box, scroll bar, and size box are not present when the X server is running. The entire window area below the Mach^Ten menu is devoted to the X desktop, or *root window*. The X desktop size automatically conforms to the physical dimensions of your screen — larger monitors will provide more surface area for your X desktop.

X client applications appear in windows on the X desktop. In Figure 27, the *xconsole*, *xterm* and Motif window manager applications are running. X client start up is discussed in section "11.2.1 Starting Clients".

11.1.2 The Menu Bar

The Mach^{Ten} menu bar appears above the X server desktop (Figure 27) and controls the X server's visibility on the desktop relative to other Mach^{Ten} terminal windows and Macintosh applications you may be running.

Unlike a Mach^{Ten} terminal window, *Command* key shortcut menu selections (the "hot" keys) are disabled when the X desktop is visible, since the Apple key is meaningful as an X *Meta* key within the X environment.

11.1.2.1 The File Menu

The File menu lets you create Mach^{Ten} terminal windows within the Mach^{Ten} window environment. The File menu is also used to quit the X Window Server environment.

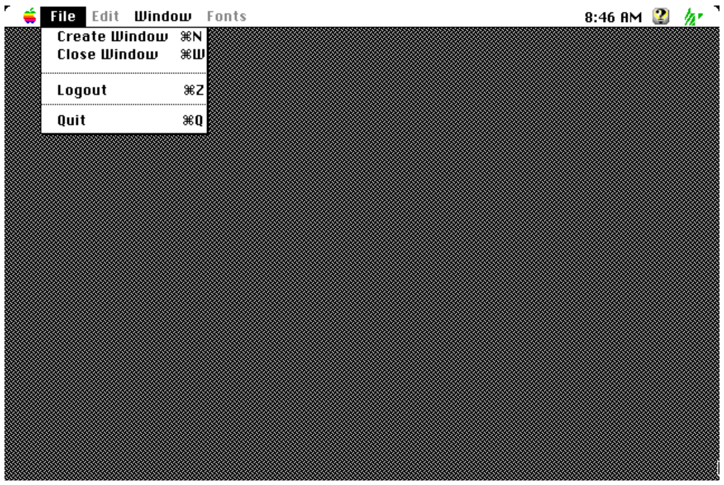


Figure 28. The File Menu

New Window	Create a new Mach^{Ten} terminal window. When a terminal window is created, the X server is hidden from view.
Close Window	Close the X server window. This option quits the X server, closing all X connections and terminating all processes associated with the X server.
Logout	Quit all Mach^{Ten} windows, including the X server, and return to the Login Console.
Quit	Quit the Mach^{Ten} application.

11.1.2.2 The Window Menu

The Window menu contains a hierarchical Order sub-menu item, a Save Window Environment item, and a list of Mach^{Ten} terminal windows including the X server. To access a sub-menu item, slide the mouse cursor over the item and to the right.

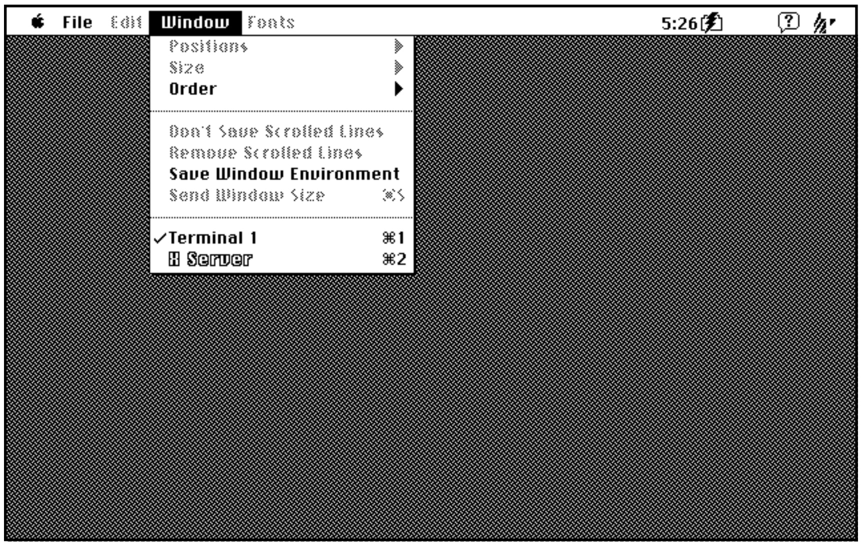


Figure 29. The Window Menu

Save Window Environment	Save a description of the sizes and locations of the current Mach^{Ten} windows environment to a named file. If the X server is running when the window environment is saved, the X server will be started automatically the next time you start up the Mach^{Ten} windows environment.
The Window List	Selecting a Mach^{Ten} terminal window makes the X desktop invisible and makes the terminal visible. Selecting *X Server* makes the X desktop visible. The active window is displayed in outline font.

The *Positions* item is disabled for all Mach^{Ten} windows when the X server is active.

The Order Sub-Menu.

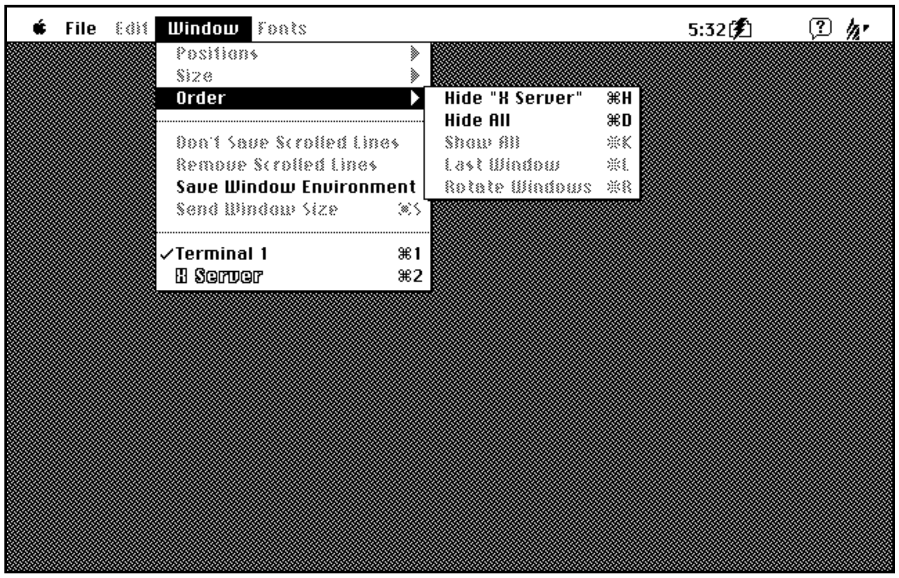


Figure 30. The Order Sub-Menu

Hide "X Server"	Make the X server invisible. The X server can be made visible using the *Show All* command or by selecting *Server* from the window list.
Hide All	Make the X server and all Mach^{Ten} terminal windows invisible.
Show All	Make the X server visible.

11.1.3 Automatic Launch of the X Server

The X server may be started automatically the next time you log into the Mach^{Ten} windows environment by selecting the *Save Window Environment* item from the Window menu. This adds a single character **lowercase** 'x' appearing on a line by itself in your Mach^{Ten} Windows environment start up file (*.windrc* by default) that indicates the X server will start up when the Mach^{Ten} Windows environment is entered.

If you close all Mach^{Ten} terminal windows (using the File menu) leaving the X server desktop as the only window in the Window menu list prior to this step, the X server will run exclusive of any Mach^{Ten} terminal window in the new environment.

11.1.4 Quitting the X Server

To quit the X server and remove the X desktop, pull down the File menu and select *Close Window*.

11.1.5 Running the X Server Without a Menu Bar

You can configure the X server desktop to cover the entire screen, including the area normally occupied by the menu bar.

When the menu bar is hidden, pull down menus from the menu bar remain accessible by the pressing the mouse button while dragging the cursor along the top edge of your X desktop.

Refer to the *-fs* X server start up option in section "11.2.3 The X Server Program" for more information on running the X server without a Macintosh menu bar.

11.1.5.1 Menu Bar Shortcuts

You can perform many of the menu bar functions described in this section using the Mach^{Ten} *xtmenu(n)* utility program. *Xtmenu* is particularly useful when you configure the X server to run without a menu bar.

Xtmenu is called from a shell command line under Mach^{Ten}. Run without options, xtmenu toggle switches the accessibility of Macintosh menu bar pull-down items on the X desktop when the menu bar is hidden.

Xtmenu also accepts the following options:

-h This option causes the visible X desktop to hide itself, exposing the Macintosh desktop. This is equivalent to selecting *Hide* from the Window Order menu.

-s When Mach^{Ten} is the foreground application and the X server is running but hidden from view, this option exposes the X desktop. This is equivalent to selecting the *X Server* item from the Window menu.

If the X server is not running, this option will start the X server, emulating the <Command-T> hot key server start up sequence.

-q When Mach^{Ten} is the foreground application, this option closes all X client connections, terminates the X server, and removes the X desktop from the Mach^{Ten} window list. This is equivalent to selecting *Close Window* from the File menu.

Window Manager Hot Keys. *Xtmenu* can be very useful when bound to a *hot* key sequence recognized by your favorite window manager. For example, you can hide the X desktop when running *mwm(n)* using the *Shift* and *Escape* keys by creating a *$HOME/.mwmrc* file with the following entry:

```
Keys DefaultKeyBindings
{
    Shift<Key>Escape    root    f.exec "xtmenu -h"
}
```

To add the same functionality to *olvwm(n)*, the *$HOME/.olvwmrc* file is modified as follows:

```
Escape + Shift {
    Execute: "/usr/bin/X11/xtmenu -h"
}
```

Window managers (with an emphasis on *mwm*) are discussed in section “11.2.2 The Window Manager Client”. Refer also to the *olvwm(n)*, *olvwmrc(n)*, and *mwm(n)* online manual pages for more information on the *.olvwmrc* and *.mwmrc* files.

11.2 Administering the X Window Software Environment

The Mach^{Ten} X Window Software is a very flexible software package that can be tailored to your individual requirements. This section describes how to install, configure and maintain the X software to create a working environment for you and your users.

If you are unfamiliar with UNIX system administration, review section “5.0 Mach^{Ten} Administration” before proceeding. An additional useful reference on administration of X is the *X Window System Administrator's Guide*, Volume 8, from O'Reilly & Associates, Inc.

11.2.1 Starting Clients

When you type <Command-T> or enter the command:

```
xtmenu -s
```

the *xinit(n)* X initialization program is launched by the Mach^{Ten} kernel. *Xinit* is responsible for starting *XMachTen(n)*, the X server program, and a default set of X clients to display on your X desktop from a start up script.

xinit may only be run by the Mach^{Ten} kernel.

11.2.1.1 The Start Up Script

The initial X clients you choose to run on your X desktop are controlled by a shell script file in your Mach^{Ten} login directory called *.xinitrc*. If *xinit* cannot find a *.xinitrc* file in your home directory, the shell script */usr/lib/X11/xinit/xinitrc* is run instead. You can copy this file to your home directory as *.xinitrc* and use it as the basis for creating your own customized X desktop. Commented lines in this file (those with a preceding pound (#) sign) are not executed by *xinit*.

The default X client start up script */usr/lib/X11/xinit/xinitrc* is shown below:

```
#!/bin/sh

userresources=$HOME/.Xresources
usermodmap=$HOME/.Xmodmap
sysresources=/usr/lib/X11/xinit/.Xresources
sysmodmap=/usr/lib/X11/xinit/.Xmodmap

# merge in defaults and keymaps

if [ -f $sysresources ]; then
    xrdb -merge $sysresources
fi

if [ -f $sysmodmap ]; then
    xmodmap $sysmodmap
fi

if [ -f $userresources ]; then
    xrdb -merge $userresources
fi

if [ -f $usermodmap ]; then
    xmodmap $usermodmap
fi

# Set the root window background

xsetroot -solid grey

# Start a console terminal

xconsole -iconic &

# Start some programs

xterm -geometry 80x24+100+20 -n `hostname` -T `hostname` &

#xclock -geometry 50x50+1-1 &

# Start a window manager

mwm

#olvwm
```

The start up script begins by calling the *xrdb(n)* client which reads in any system-wide resources defined in the file */usr/lib/X11/xinit/.Xresources*, if present. Resources that you want available to all users and all clients displaying to the server should be defined in this file. Resources will be discussed in more detail in a subsequent section.

Next the script runs the *xmodmap(n)* client to load a custom keyboard mapping file from */usr/lib/X11/xinit/.Xmodmap*, if present. If you are using a U.S. keyboard, you will normally have no need to change your keyboard mapping since the X server is launched by default with a U.S. keyboard mapping. If you operate the X server using a non-U.S. keyboard, the *.Xmodmap* file should be linked to one of the international keyboard map files in */usr/lib/X11/xinit*. Details on using the international keyboard map files are found in section "11.2.3.3 Keyboard Mapping".

The start up script runs *xrdb* and *xmodmap* again using *.Xresources* and *.Xmodmap* files from your home directory. These files can be maintained in individual user home directories to customize the default server resources or key map for that user.

The start up script uses *xsetroot(n)* to change the background, or root window, to a grey color. Choose any color from the list of colors in */usr/lib/X11/rgb.txt*.

Next, *.xinitrc* starts a console X window as an icon on the X desktop. Mach^{Ten} console output is redirected to the *xconsole(n)* window and is visible when the icon is *de-iconified*, or opened with a double-click of the mouse button.

Next, *.xinitrc* starts the X applications, by default an *xterm(n)* virtual terminal window. The window's geometry, title and icon name corresponding to the local host name are provided as command line parameters when starting the *xterm* client. You can add to the list of clients to be run at this point in the script. Note that the *xclock(n)* client is not run by default because of the comment ('#' sign) preceding the command. Also note that each application is placed in the background by the shell (using the '&' sign) to allow the entire client list to be processed.

Finally, the start up script launches a special X application known as the *window manager* that allows you to move, resize, and iconify your X application windows. By default, *mwm(n)*, the Motif Window Manager, is launched. Also available in the Mach^{Ten} X Window Software is *olvwm(n)*, the OpenLook Virtual Window Manager, and *twm(n)*, the Tab Window Manager, from the MIT sources distributed with the X Window Software package. The operation of the Motif Window Manager is described in detail in section "11.2.2 The Window Manager Client".

The start up script you create in *.xinitrc* in your home directory takes precedence over the default system start up script.

11.2.1.2 Resources: X Application Preferences

Macintosh applications permit the setting of preferences to customize the application to the individual user's tastes, for example background color, window title, font size, etc. Preferences are set for the application and typically stored in a single file in the Macintosh Preferences folder.

Under X, preferences are set via application *resources*. Resources are specified as strings that are read in from multiple source files when an application is run. Application resources under X are inherently more flexible than Macintosh application preferences due to the client / server model under which X operates. As with Macintosh programs, preferences can be specified for the individual X application. In addition, since Mach^{Ten} allows multiple simultaneous users, X application preferences can also be specified for each individual user. Since the same resource source directory may be shared by several different hosts (for example a NFS file server serving a user's login directory to multiple hosts), the user's application preferences can be specified for the specific host. Similarly, because X clients can display on any networked X server and each server may need special preferences, resources can be defined specific to the X server.

Resource Files. Resource files contain the default application resources and are commonly provided with the X application you are running. The resource file is named after the *class* name of the application. By convention, the class name of the application is the same as the program name, but with the first letter capitalized (for example Bitmap or Emacs) although some programs that begin with the letter 'x' also capitalize the second letter. Application class names can be discovered using the *xprop(n)* utility.

Resource Syntax. In their simplest form, resources have the following syntax:

```
name*variable:value
```

For example:

```
XTerm*background:beige
```

where *name* is either the class name or the *instance* name of the application. The default instance name is the name of the program, for example:

```
xterm*background:blue
```

Instance names may be changed using the *-name* command line option when you start the application. You can use the *-name* option to distinguish more than one instance of the same application in your resource file.

As an example, suppose you want *xterm* windows from host *ray* to have a blue background and from host *fred* to have a red background. Your resource file may contain the following resources:

```
xterm-ray*background:blue
xterm-fred*background:red
```

When you start the *xterm* client from hosts ray and fred, use the *-name* option to match the instance name used in the resource:

```
xterm -name xterm-ray &       (xterm started from ray)
xterm -name xterm-fred &      (xterm started from fred)
```

instance names take precedence over *class* names in resource specifications.

Applications written with the X Toolkit Intrinsics will have at least the following resource variables::

background	This resource specifies the color to use for the window background.
foreground	This resource specifies the color to use for the text and graphics within the window.
borderWidth	This resource specifies the width in pixels of the window border.
borderColor	This resource specifies the color to use for the window border.

Resource File Locations. Programs based on the X Toolkit Intrinsics obtain resources from the following sources. The sources are listed in the order of precedence from lowest to highest precedence: if the same resource specification is located in more than one place, the resource in the source with the higher precedence is used.

Application Level Resources.

- **/usr/lib/X11/app-defaults**

  This directory is akin to the Macintosh Preferences folder. This is a directory of default application resource files. Within each file, for example *XTerm*, resource variables are defined for the application. When started, the application consults only the file in this directory corresponding to the class name of the application.

User Level Resources.

- **$HOME/<application file>**

  The application default resource file may be copied to your home directory and customized according to your individual preferences. The file *~/XTerm* will take precedence over */usr/lib/X11/app-defaults/XTerm* when *xterm* is started.

- **XAPPLRESDIR**

  This is an environment variable you can set to an alternate search path for your application resource files; for example:

  ```
  setenv XAPPLRESDIR /tmp/app-defaults
  ```

  causes the X client to search for its resource file from an *app-defaults* subdirectory under the */tmp* directory.

- **$HOME/.Xdefaults**

  If the X server database (described below) contains no resources, resources are read from a *.Xdefaults* file in your home directory when the application starts up. This file is not specific to an application and may contain resources from any number of different X clients.

X Server Level Resources.

- **X Server Database**

  Any global resources that should be available to all clients on all machines can be stored in a database residing on the X server using the *xrdb(n)* program. This is typically done when the server starts up in the *.xinitrc* file in the user's home directory. If the file *usr/lib/X11/xinit/.Xresources* containing server level resources is found, the resources are loaded into the X server and automatically become available to all client applications from all systems, including the local host system. The X server database is loaded using the command:

  ```
  xrdb -merge <resource file>
  ```

  If any resource is loaded into the X server database via *xrdb* (including non-conflicting resources), the *$HOME/.Xdefaults* file will not be read.

Host System Level Resources.

- **$HOME/.Xdefaults-<hostname>**

  This file is associated with the host (from the *hostname(1)* command) on which you are executing the X application. It is useful when you share a single NFS mounted home directory among multiple hosts. It takes precedence over all application, user, and server level resources previously mentioned. It is not specific to an application and may contain resources for any number of different X applications.

- **XENVIRONMENT**

  If this environment variable is set to a named file, the file is read instead of the *.Xdefaults-<hostname>* file. It takes precedence over all other previously mentioned resource locations when starting an X application, for example:

  ```
  setenv XENVIRONMENT ~/.Xdefaults-custom
  ```

  The named file typically contains custom resources for all of the applications you run.

Command Line Resource Specification.

- **-xrm *resourcestring***

  Resources can also be specified from the command line when starting your X application. The *resourcestring* is a single resource name, variable and value.

  For example, the *xterm* default background color is overridden with a red background using the command:

  ```
  xterm -xrm "XTerm*VT100.background: red"
  ```

  Note that if the string contains characters interpreted by the shell (e.g., asterisk), they must be quoted. Any number of *-xrm* arguments may be given on the command line.

- **Command Line Options**

  A command line option to the X application takes ultimate precedence over all other resource locations. An *xterm*'s background will be blue regardless of all *xterm* background resource specifications in all locations when *xterm* is started with the command:

  ```
  xterm -bg blue
  ```

Recommendations for Administering Resources. Maintain unmodified copies of application resources in the */usr/lib/X11/app-defaults* directory. Customize these resources for individual users by copying the customized variables from each resource file to a *.Xdefaults* file in the user's home directory.

If you run the same client from multiple hosts and you want to retain a consistent appearance for the client, consider using *xrdb* to load the client's resources directly into the X server. This method eliminates the need to maintain multiple identical copies of your *.Xdefaults* file on each host.

If in doubt as to which resources an X application will use when it is launched, use the *appres(n)* program to preview the application's resources.

For more information on the resource variables available to an application, refer to the online manual page for the application.

11.2.2 The Window Manager Client

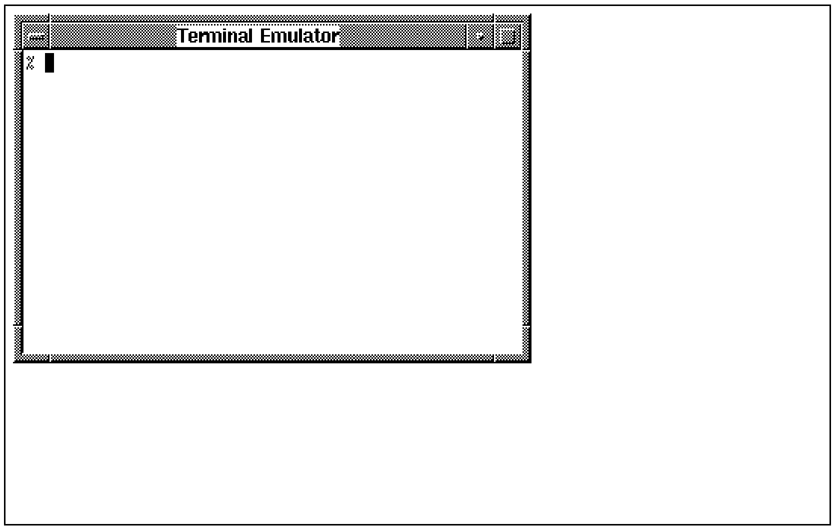
The *window manager* is a program that runs with the X Window System and helps you manage the windows on your screen. The window manager provides functions for opening, closing, moving, and resizing windows. It's the window manager that displays the three-dimensional frame around each window. Without a window manager, your windows would not have borders and it would be difficult, if not impossible, to perform many simple window functions.

The Mach^{Ten} X Window Software provides you a choice of *olvwm(n)*, the OpenLook Virtual Window Manager, *mwm(n)*, the Motif Window Manager, and *twm(n)*, the Tab Window Manager. This section will discuss the operation of *mwm*. Refer to the online manual pages for more information on *olvwm* and *twm*.

11.2.2.1 Starting the Window Manager

Your default client start up script (*.xinitrc* in your home directory) launches the window manager when the X server is started. In combination, the X server, your X applications, and the window manager comprise the X desktop.

The resources for the *xterm* client govern the precise size and appearance of the terminal emulator window shown in the figures in this section.



```
% 
```

Figure 31. The Motif Environment

11.2.2.2 Using the Mouse

As you move the mouse on your X desktop, the pointer on the screen moves correspondingly.

Whenever this section tells you to "point to" something, it simply means move the mouse until the pointer on your screen is positioned over the "something."

Pointer Shapes. The location of the pointer can cause the shape of the pointer to change. For example, when the pointer is directly over the root window (the backdrop behind all windows), the pointer has an arrow shape. When you point to the inside of a terminal window, the pointer changes to an "I" shape. See Figure 32.

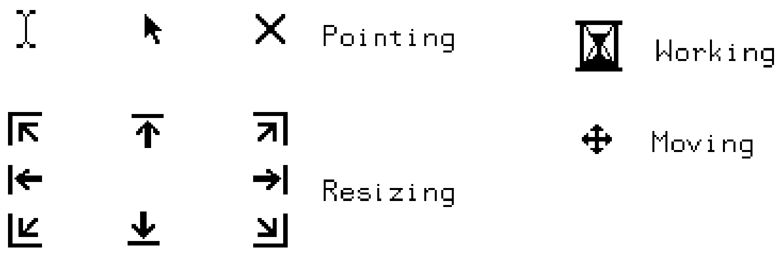


Figure 32. Motif Pointer Shapes

Clicking, Double-Clicking and Dragging. The following terminology is used to refer to actions involving mouse buttons:

- "Click" means to press and release a button without moving the pointer
- "Double-click" means to click a button twice in rapid succession
- "Drag" means to press and hold a button while moving the pointer

11.2.2.3 Selecting a Window

Before a window can receive input from you, it must be selected as the active window. To select a window, point to any part of the window and click the select button on the mouse.

Select the terminal window on your screen now. Point to the window and then click the select button. (The select button is your mouse button or the left button on a three button mouse.)

When a window is active, the window frame changes appearance. Now, when you press on the keyboard, the characters appear on the command line in the active window.

If no window is active, everything you type is lost.

11.2.2.4 Moving a Window

Using the pointer, you can move a window to a new location on the screen. To do so, grab the window's title bar with the pointer, drag it to a new location, and then release the window.

Step 1 — Grabbing the Title Bar With the Pointer

Point to the title bar of the window. The title bar is the rectangular area across the top of the window where the words *Terminal Emulator* appear. When properly positioned, the pointer is an arrowhead as shown in Figure 33.

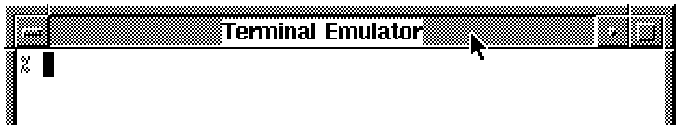


Figure 33. Grabbing the Title Bar

When the pointer is positioned, press and hold the select button.

Step 2 — Dragging the Window to a New Location

Still holding the select button, slide the mouse to the right across your desktop. Now slide the mouse toward you.

As you move the mouse, the pointer on the screen drags an outline of the terminal window as shown in Figure 34. The outline shows you where the window will be moved when you release the mouse button.

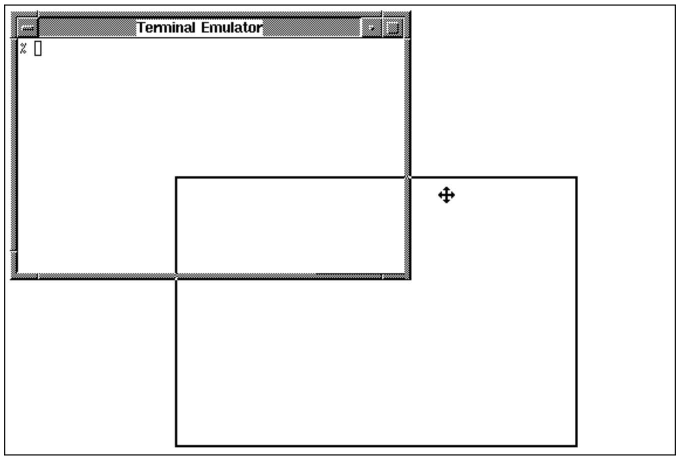


Figure 34. Repositioning the Window

Step 3 — Releasing the Window

Move the outline to the center of the screen. When the outline is where you want it for now, release the mouse button. The window moves to the new position.

11.2.2.5 Changing the Size of a Window

You can change the size of a window by grabbing an area of the window's frame with the pointer, dragging the frame to the desired size, and then releasing the frame.

Step 1 — Grabbing the Frame With the Pointer

Where you grab the window's frame determines how the window will be resized. See Table 6.

Table 6: Places to Grab on the Window Frame

If you want to stretch or shrink the window ...	Point to ...
vertically from the ...	
top	top of the frame, above the title bar
bottom	bottom of the frame
horizontally from the ...	
right	right side of the frame
left	left side of the frame
diagonally from the ...	
bottom left corner	frame's lower left corner
top left	frame's upper left corner
top right	frame's upper right corner
bottom right	frame's lower right corner

For this exercise, point to the lower right corner of the window frame. When positioned correctly, the pointer's shape changes to an arrowhead pointing into the corner of the frame. Grab the frame by pressing and holding down the select button.

Step 2 — Enlarging or Shrinking the Window

With the select button still pressed, slide the mouse so that the pointer moves away from the window. Now slide the mouse so the pointer moves toward the window. As you move the mouse, the pointer on the screen drags an elastic outline of the window. The outline shows you the new size of the window.

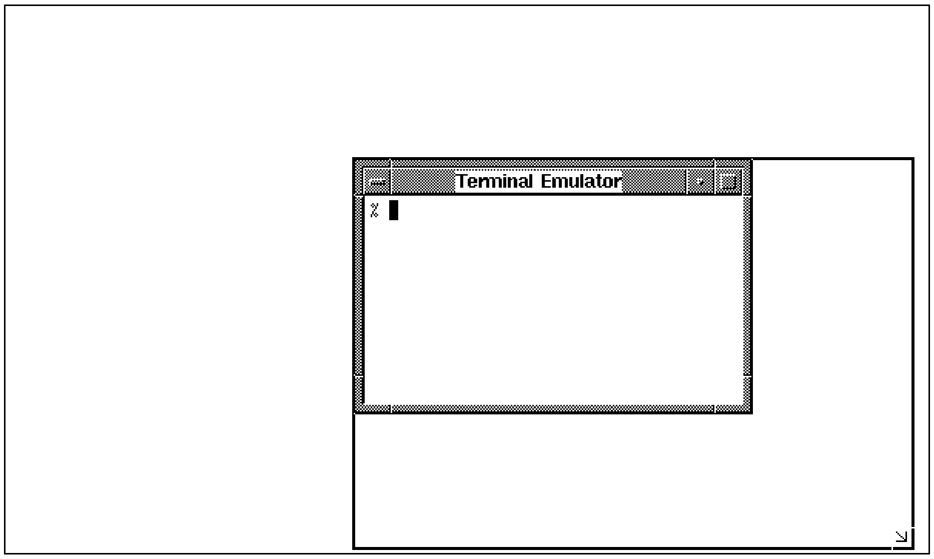


Figure 35. A Window and Its Elastic Outline

Now, stretch the window until it reaches the lower right corner of the root window as shown in Figure 35.

Step 3 — Releasing the Frame

When the elastic outline is the correct size, release the mouse button. The screen is redrawn with the window filling the outline.

11.2.2.6 Changing a Window Into an Icon

As you work, your screen can become cluttered with windows. Changing a few of those windows into icons allows you to tidy up a cluttered workspace. Icons are small graphical images. Programs executing in a minimized window continue to execute until they finish or halt because they require input from you.

Step 1 — Locating the Minimize Button

Slide the mouse so that you position the pointer on the minimize button — the small square to the immediate right of the title bar on the window frame as shown in Figure 36.

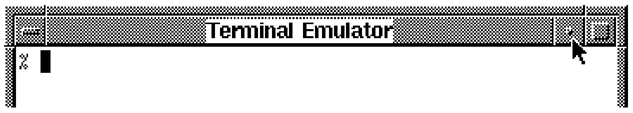


Figure 36. The Title Bar Minimize Button

Step 2 — Pressing the Minimize Button

Press the minimize button by clicking the select button.

The window changes into an icon. *Mwm* displays icons in a row along the bottom of the root window, starting on the left. An icon has two parts, as shown in Figure 37.

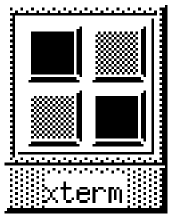


Figure 37. The Motif Icon

11.2.2.7 Moving an Icon

To move an icon around the screen, follow these steps:

1. Point to the icon.
2. Press and hold the select button.
3. Still holding the select button, drag the pointer. An outline follows the pointer, showing where the icon will be repositioned.
4. To position the icon, release the mouse button.

Icons are positioned on a grid, so the icon you move may jump a bit if you set it too close to the margin of the root window or too close to another icon.

11.2.2.8 Restoring a Window From an Icon

To restore an icon (change it back into a window), point to the icon and double-click the select button. To double-click a mouse button, press the button twice in rapid succession.

Step 1 — Pointing to an Icon to Restore

Point to the icon by positioning the mouse pointer in the icon. If you can't identify the icon you want to restore because some icon names are obscured within the icons' title areas, you can display an entire label by clicking the select button in an icon's title.

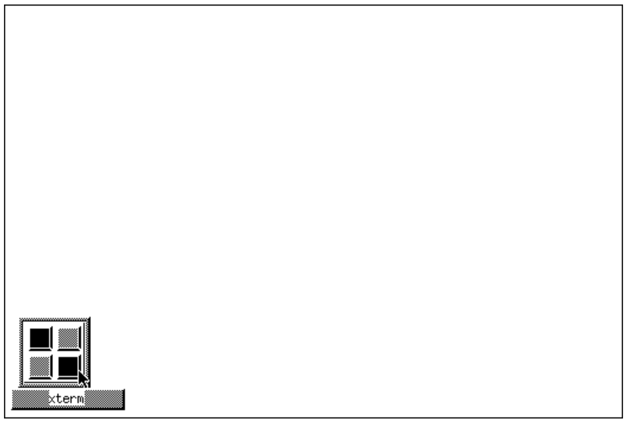


Figure 38. An Icon and Its Pop-up Label

Step 2 — Changing the Icon Into a Window

While pointing to the icon, double-click the select button. That is, press the left mouse button twice in rapid succession.

The icon changes back into the window it originally came from. The window positions itself where it was before being minimized, as shown in Figure 39.

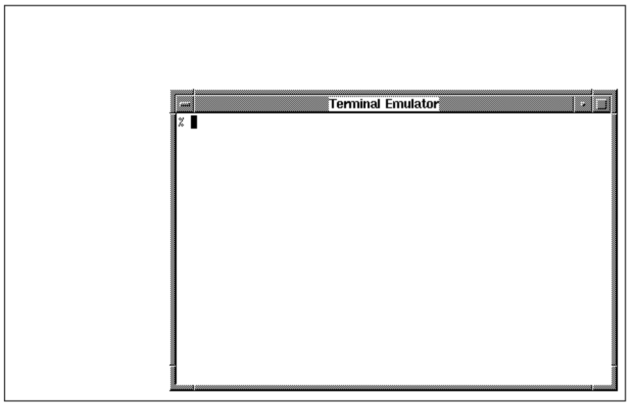


Figure 39. The Restored Window

11.2.2.9 Displaying a Window Menu and Making Selections

Each window has a window menu that contains functions for controlling the window.

Step 1 — Displaying the Window Menu

The window menu is attached to the window frame and you use the menu button in the upper-left corner of each window to display it.

To display the window menu, point to the menu button and then press and hold the select button. The menu is displayed as long as you hold the button down — don’t release the button yet. See Figure 40.

Step 2 — Choosing a Function From the Menu

While still holding the select button down, drag the pointer down the menu. As the pointer moves, it highlights the button for each available selection. Drag the pointer until you highlight the Maximize function. Release the mouse button.

The Maximize function causes the window to expand to fill the entire screen.

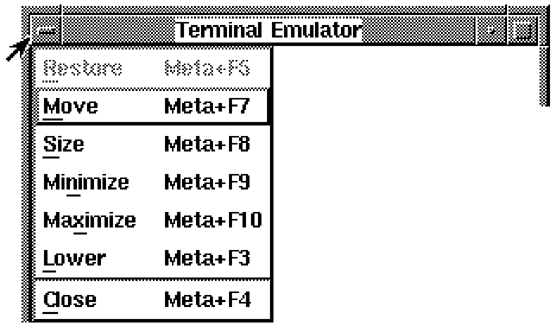


Figure 40. The Motif Window Menu

Step 3 — Restoring the Window to its Original Size

Display the window menu again (point to the window menu button in the upper-left corner of the window and press the select button). Drag the pointer down the menu until you highlight the Restore selection. Release the mouse button. The window is restored to its former size and location. Table 7 lists the window menu functions.

11.2.2.10 Summary of Window Menu Functions

Table 7: Functions in the Window Menu

To do this ...	Choose ...
Restore a window from an icon or after maximizing.	Restore
Change the location of the window.	Move
Change the size of the window.	Size
Shrink the window to its icon representation.	Minimize
Enlarge the window to cover the entire root window.	Maximize
Send a window to the back or bottom of the window stack, the position closest to the root window.	Lower
Immediately stop the window and make it disappear	Close

When a menu function is meaningless, its name is grayed out and you cannot select it. For example, the Restore function in Figure 40 is inactive. Rather than choosing the Maximize and Restore functions in the previous example, you could have used the Maximize button in the upper right corner of the window, immediately to the right of the Minimize button.

11.2.2.11 Raising a Window

To bring a partially concealed window to the front of the root window, click on the window's frame.

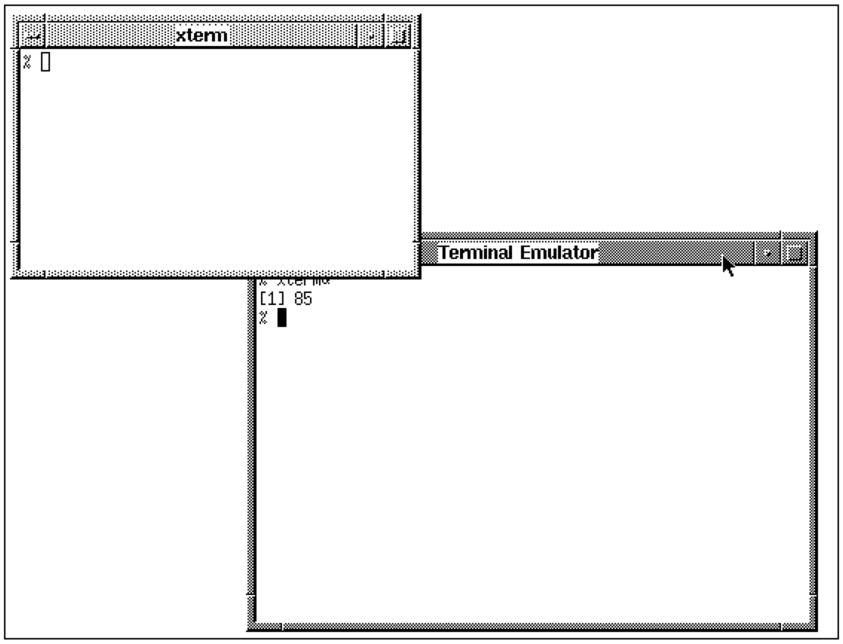
To demonstrate this, create another terminal window by typing this command in the existing terminal window:

```
xterm & <CR>
```

The new window appears in the upper-left corner of the display, partially covering the first window as shown in Figure 41.

Step 1 — Positioning the Pointer

Position the pointer on any visible portion of the concealed window's frame.



```
% 
```

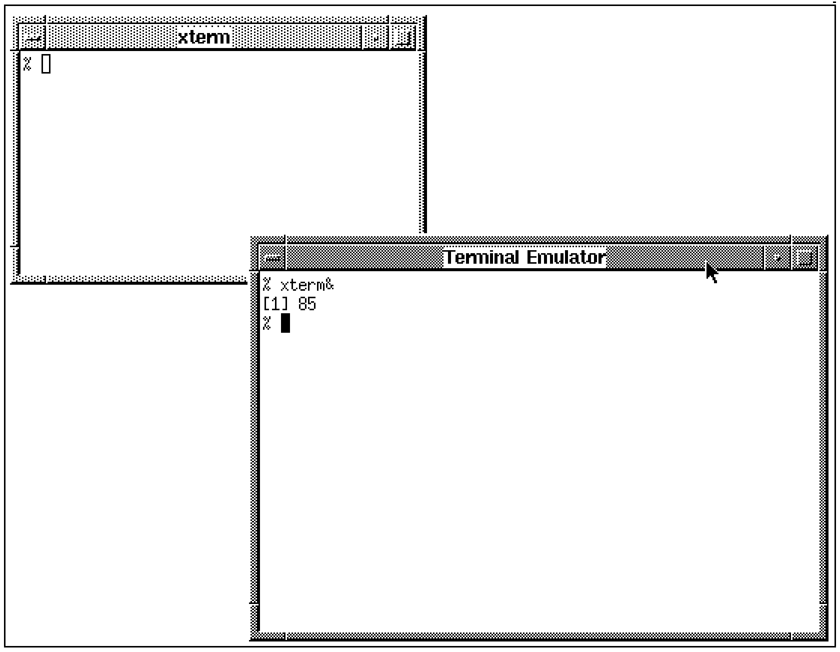
```
% xterm&
[1] 85
% 
```

Figure 41. A Partially Concealed Window

Step 2 — Pressing the Select Button

Click the select button on the mouse.

The partially concealed window moves to the front of all other windows as shown in Figure 42.



```
% []
```

```
% xterm&
[1] 85
% 
```

Figure 42. The Window Revealed

11.2.2.12 Quitting the Window Environment

Before exiting the window environment, exit any application programs and stop any commands that may be running in terminal windows.

Pull down the File menu item from the Mach^{Ten} menu bar and select *Close Window*. This removes from the screen all terminal windows (as well as the windows for clocks and similar windowed programs) and terminates the X server.

11.2.3 The X Server Program

When the X server starts up, it takes over the display, except for the Macintosh menu bar running horizontally along the top of the display. Use the menu bar to suspend the X server to access the Mach^{Ten} terminal window environment or other running Macintosh applications.

The X server program is the file */usr/bin/X11/XMachTen* or by its linked name, */usr/bin/X11/X*. By default, *xinit(n)* will automatically run */usr/bin/X11/X*.

You are prevented from running the X server program except by way of *xinit* under Mach^{Ten}.

11.2.3.1 X Server Start Up Options

By default, *xinit* will automatically run */usr/bin/X11/X*. Alternatively, if a file named *.xserverrc* is found in your home directory, *xinit* will execute this file as a shell script instead. The *.xserverrc* file is used if you need to start the X server with options. Start up options alter the default X server behavior and are useful for three button mouse operation, font path redirection and alternate server display number operation, among other things.

A sample X server start up script is in */usr/lib/X11/xinit/xserverrc*. Copy the file to your home directory as *.xserverrc* if and only if you want to start the server with options. Note that all lines in the sample file are commented out, so if you decide to use a *.xserverrc* file (by placing one in your home directory), make sure a single line that starts the server with the options you want is uncommented. If no *.xserverrc* file is found in your home directory, *xinit* starts the X server without options.

The X server program accepts the following command line options:

-ac	Disables host-based access control mechanisms. Enables access by any host, and permits any host to modify the access control list. Use with extreme caution. This option exists primarily for running test suites remotely.
-auth *authorization-file*	Specifies a file which contains a collection of authorization records used to authenticate access. See also the *xdm* and *Xsecurity* manual pages.
bc	Disables certain kinds of error checking, for bug compatibility with previous releases (e.g., to work around bugs in R2 and R3 *xterms* and toolkits).
-bs	Disables backing store support on all screens.
-co *filename*	Sets name of RGB color database.
-dpi *resolution*	Sets the resolution of the screen, in dots per inch. To be used when the server cannot determine the screen size from the hardware.

-fc *cursorFont*	Sets the default cursor font.
-fn *font*	Sets the default font.
-fp *fontPath*	Sets the search path for fonts. This path is a comma separated list of directories which the X server searches for font databases. It may also indicate a network font server in the form: `tcp/`**`hostname`**`:7000` where *hostname* is the machine name of the network font server.
-fs	Starts the server in full screen mode. In this mode, the Macintosh menu bar is hidden by the X desktop. Pull-down menus from the Macintosh menu bar remain accessible by pressing the mouse button while dragging the cursor along the top edge of the desktop.
-help	Prints a usage message.
-I	Causes all remaining command line arguments to be ignored.
-ld *kilobytes*	Sets the data space limit of the server to the specified number of kilobytes. A value of zero makes the data size as large as possible. The default value of -1 leaves the data space limit unchanged.
-ls *kilobytes*	Sets the stack space limit of the server to the specified number of kilobytes. A value of zero makes the stack size as large as possible. The default value of -1 leaves the stack space limit unchanged.
-mbm *middle-key*	Changes the middle mouse button keystroke mapping. The key entered as a decimal value representing the Macintosh virtual key code corresponding to the desired key. Allowable values are between 0 and 127. Refer to section “11.2.3.2 Mouse Button Mapping” for a table of keys and virtual key codes.
-mbr *right-key*	Changes the right mouse button keystroke mapping.
-menu	Starts the server in *dedicated* full screen mode. In this mode, the Macintosh menu bar is hidden by the X desktop and access to Macintosh pull-down menus is completely disabled. The *xtmenu(n)* program, run from your X window manager menu environment, may be used to temporarily restore access to your Macintosh desktop. Caution: this option locks out access to Finder and other running Macintosh applications!

-mo	Specifies that the Macintosh Option key will be used in combination with the mouse button keystroke to simulate the middle or right mouse button. When used without the *-mbm* or *-mbr* start up options, the default middle and right mouse button keystrokes become <Option-left arrow> and <Option-right arrow>, respectively. Selecting this option causes all arrow keys, including the Up and Down arrows, to function normally.
-static	Runs the server in StaticColor mode, preventing X client modification of the Macintosh color table.
-su	Disables save under support on all screens.
-to *seconds*	Sets the default connection time-out in seconds.
-wm	Forces the default backing-store of all windows to be *WhenMapped*; an easy way of getting backing-store to apply to all windows. This option can be useful when displaying X applications over slow serial links.

You can also have the X server connect to an X display manager using XDMCP, the X Display Manager Control Protocol. Refer to section "11.2.5 X Display Management under Mach^{Ten}" for more information on the X display manager. The following options control the behavior of XDMCP:

-query *host-name*	Enable XDMCP and send *Query* packets to the specified host.
-broadcast	Enable XDMCP and broadcast *BroadcastQuery* packets to the network. The first responding display manager will be chosen for the session.
-indirect *host-name*	Enable XDMCP and send *IndirectQuery* packets to the specified host.
-port *port-num*	Use an alternate port number for XDMCP packets. Must be specified before any *-query*, *-broadcast* or*-indirect* options.
-once	Normally, the server keeps starting XDMCP sessions, one after the other. This option allows the server to continue operating without XDMCP access and control restrictions after a single XDMCP session.
-class *display-class*	XDMCP has an additional display qualifier used in resource lookup for display-specific options. This option sets that value, by default it is *MITUnspecified.*
-cookie *xdm-auth-bits*	When testing XDM-AUTHENTICATION-1, a private key is shared between the server and the manager. This option sets the value of that private data.
-displayID *display-id*	An XDMCP specific value allowing the display manager to identify each display so that it can locate the shared key.

11.2.3.2 Mouse Button Mapping

Many X clients assume the mouse has three buttons. The X server simulates the middle and right mouse buttons with keystrokes — by default the left-arrow key generates middle button events and the right-arrow key generates right button events — the real mouse button generates left button events. The open-Apple or Command key is the *Meta* (or when shifted, the *Alt*) modifier; Meta can also be obtained by pressing the up-arrow key. The down-arrow key duplicates the Control key. Meta, Control and Shift are often used in combination with other keystrokes or mouse clicks. For example, the terminal emulator *xterm* pops up menus in response to Control-middle and Control-right.

The original function of the arrow keys may be obtained by holding down the Option key while pressing one of the arrow keys. Normal arrow key functions are also restored when you re-map the middle and right mouse buttons using the *-mbm* and *-mbr* server start up options or when you configure the server to accept the <Option-mouse key> combination using the *-mo* start up option.

Three Button Mouse Configuration. Several commercial three button mice are offered for the Macintosh. All operate by assigning an unused keyboard key or key combination to the middle and right mouse buttons. The X server supports the reconfiguration of the mouse keys from the default left-arrow and right-arrow to any other keyboard key or <Option-key> combination. By matching the X server mouse button configuration with your three button mouse hardware configuration, you can shift your X three button operation entirely to the hardware mouse.

The default keystrokes for the middle and right mouse buttons may be changed via the *-mbm* and *-mbr* server start up parameters, described in the previous section. The values to assign to the options for a given Macintosh key are as follows:

Key	Value	Key	Value
F1	122	Help	114
F2	120	Home	115
F3	99	Page up	116
F4	118	Page up	116
F5	96	Page down	121
F6	97	End	119
F7	98	Left arrow	123
F8	100	Right arrow	124
F9	101	Down arrow	125
F10	109	Up arrow	126
F11	103	Clear	71

Key	Value	Key	Value
F12	111	Equal	81
F13	105		
F14	107		
F15	113		

The X server may also be configured to accept the Macintosh *Option* key in combination with the selected button keystroke using the -*mo* start up parameter described in the previous section. This allows keyboard keys to function both as normal keys and, when the *Option* key is pressed, as mouse buttons.

If the default mouse button configuration is modified or you use the Macintosh *Option* key in combination with another key to simulate a mouse button, all arrow keys revert to their original functions.

Example Three Button Mouse Configuration. Suppose you own a Logitech Mouse and would like to map the middle and right mouse buttons to the <Option-F14> and <Option-F15> key combinations when running the Mach^{Ten} X server. Your Logitech MouseMan™ Control Panel would look something like this:

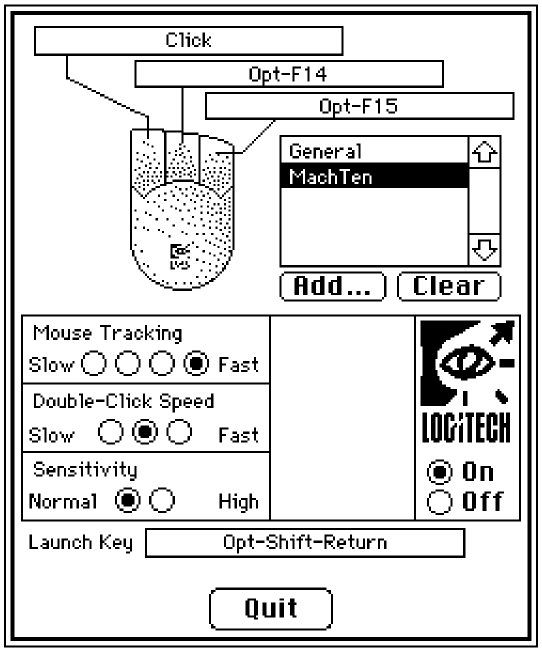

Click
Opt-F14
Opt-F15
General
MachTen
Mouse Tracking: Slow ○ ○ ○ [✓ ●] Fast
Double-Click Speed: Slow ○ [✓ ●] ○ Fast
Sensitivity: Normal [✓ ●] ○ High
[✓ On] Off
Launch Key: Opt-Shift-Return

Figure 43. Logitech MouseMan™ Control Panel

The *.xserverrc* server start up file in your home directory would look like this:

```
#!/bin/sh
X -mo -mbm 107 -mbr 113
```

The */usr/lib/X11/xinit/xserverrc* sample X server start up script shows another example of starting the X server with a three button mouse configuration.

11.2.3.3 Keyboard Mapping

The default layout for keyboard keys in the X server corresponds to the U.S. character set under Mac OS. Special characters and symbols, including characters with diacritical marks, are available by pressing the Option key in combination with certain letter and number keys. Refer to the *Macintosh User's Guide* for more information on typing special characters.

Configuring International Keyboard Layouts. Your Keyboard control panel indicates the keyboard layout in force for your Macintosh. If your keyboard layout is an international (non-U.S) layout, the default U.S. X key layout must be re-mapped when the X server is launched to match your country's layout.

The *xmodmap(n)* program is used to re-map the default U.S. character set within the X server for use in other countries. *Xmodmap* is run by the *.xinitrc* script when the X server starts up. The script searches for a key mapping file called *.Xmodmap* in the directory */usr/lib/X11/xinit*. If found, *xmodmap* configures the X server using the contents of the mapping file.

Mapping files are named *Xmodmap-XX* where *XX* represents the country code for the desired keyboard layout:

DE	Germany
DK	Denmark
ES	Spain
FI	Finland
FR	France
GB	Great Britain
IT	Italy
NO	Norway
PT	Portugal
SE	Sweden
FC	French Canadian
CA	Canada

To change the X default U.S. keyboard layout, enter the following from a Mach^{Ten} terminal window prior to starting the X server for the first time:

```
ln /usr/lib/X11/xinit/Xmodmap-XX /usr/lib/X11/xinit/.Xmodmap
```

This command creates a *.Xmodmap* link file to your country's X key map. The *.Xmodmap* file will be read into the server by *xmodmap* each time the server starts up via the *.xinitrc* file. To restore the X key map to the default U.S layout, delete the link file and restart the X server.

For international X keyboard map files not listed above, contact technical support at Tenon Intersystems.

X Meta and Alt Keys. Under the Mach^{Ten} X Server, hot key operation (using the Apple Command key) normally associated with a Mach^{Ten} terminal window is disabled. The Command key functions as an X *Meta* (or when shifted, *Alt*) key when the X server is the front window. Many applications, including the Motif Window Manager, use the *Meta* and *Alt* keys as keyboard modifiers.

Apple reserves certain keystroke combinations for special effects. All *Command-Shift-number* sequences are not delivered to the X server and are unavailable for use by your X applications.

The Macintosh Option key is not available for use as a Meta key.

11.2.3.4 Controlling X Client Access to the Server

In its default mode of operation, the X server uses a host-based access control list for deciding whether or not to accept connections from clients on a particular machine. If no other authorization mechanism is being used, this list initially consists of the host on which the server is running, as well as any machines listed in the file */etc/X0.hosts*. Each line of the file should contain an Internet host name. There should be no leading or trailing spaces on any lines. For example:

```
sunworkstation
division.company.com
```

You can add or remove hosts from this list and enable or disable access control using the *xhost(n)* command from the same machine as the server.

11.2.3.5 Using an Alternate Server Display Number

The server listens for X connections on TCP port 6000+n, where n is the display number. The default display number is zero (0).

X clients (run locally or from the network) access your X server by connecting to the server on the display number. This is indicated by the DISPLAY environment variable for the shell running the X client:

`:0`	(for local connections)
`<yourhostname>:0`	(for network connections)

You can configure the X server to listen for connections on an alternate display by adding the parameter *:n* to the X server start up command line in your *.xserverrc* file, where *n* is the alternate display number. You must also change the DISPLAY variable in your local Mach^{Ten} *.login* or *.profile* file (in the home directory of the account starting the X server) to match the alternate display number.

11.2.3.6 Server Error Logging

The X server logs cumulative error messages to the file */var/adm/X0msgs*. If you are having trouble starting the X server, or if the server suddenly exits, the contents of this file may contain helpful troubleshooting information.

If the *X0msgs* file does not exist, the X server will create a new one when it starts up.

You may optionally remove the *X0msgs* file and create a symbolic link to your console terminal:

```
ln /dev/console  /var/adm/X0msgs
```

With the symbolic link in place, errors logged by the X server will appear in the Mach^{Ten} console window.

11.2.3.7 X Server Performance Tuning Guide

The following tips will help you optimize the performance of the Mach^{Ten} X Server.

- For best performance displaying local X client applications launched from a Mach^{Ten} terminal window, the DISPLAY environment variable in your *.login* or *.profile* file should be set to *unix:0* or simply *:0*. Sample *.login* and *.profile* files showing the DISPLAY environment variable syntax are in the */usr/share/skel* directory.

- Turn off Apple File Sharing when not in use.

- Close all open Control Panels on your Macintosh desktop.

- Macintosh and X screen saver client applications will conflict with one another when run simultaneously. Hide the X server and run your favorite Macintosh screen saver package to protect your screen during idle periods; or run an X screen saver application (e.g. *xlock*) with your Macintosh screen saver disabled. *Xlock* is available as a ported Mach^{Ten} application on *ftp.tenon.com* in the */pub/applications/X11* directory.

- When using MultiFinder to switch from the X desktop to Finder, a Mach^{Ten} terminal window, or other Macintosh applications, iconify any X clients that are actively drawing to the display prior to hiding the X desktop.

- Mach^{Ten} minimizes the impact of idle Macintosh applications (including Finder) on X server performance; however, for the UNIX enthusiast, running Mach^{Ten} as the only application (masquerading as Finder) will provide an additional performance boost. A Tenon technical note on running Mach^{Ten} as Finder is available at *ftp.tenon.com* in */pub/tech_notes*.

11.2.4 Font Service

A font is a collection of text characters and symbols in a particular style. All X servers, including X terminals, must have access to fonts to display text. Fonts are available to X servers either from local storage or from a network *font server*.

Under Mach^{Ten}, you can configure the X server to access fonts from the local disk (the default configuration) or from a network font server. Your Mach^{Ten} system may also be configured to serve fonts to other Mach^{Ten} X servers as well any X11R5 based X window display server on your network.

11.2.4.1 The Default Font Path

X fonts are stored locally as individual files in directories. The list of directories and font servers an individual X server uses when trying to open a font is controlled by the *font path*.

The default font path for the X server contains four directories:

/usr/lib/X11/fonts/misc	This directory contains many miscellaneous bitmap fonts that are useful on all systems. It contains a family of fixed-width fonts, several Kana fonts from Sony Corporation, two JIS Kanji fonts, two Hangul fonts from Daewoo Electronics, two Hebrew fonts, the standard cursor font, two cursor fonts from Digital Equipment Corporation, and cursor and glyph fonts from Sun Microsystems. It also has various font name aliases for the fonts, including fixed and variable.
/usr/lib/X11/fonts/Speedo	This directory contains outline fonts for Bitstream, Inc.'s Speedo rasterizer. A single font face, in normal, bold, italic, and bold italic, is provided.

/usr/lib/X11/fonts/75dpi	This directory contains bitmap fonts contributed by Adobe Systems, Inc., Digital Equipment Corporation, Bitstream, Inc., Bigelow and Holmes, and Sun Microsystems, Inc. for 75 dots per inch displays. An integrated selection of sizes, styles, and weights are provided for each family.
/usr/lib/X11/fonts/100dpi	This directory contains 100 dots per inch versions of some of the fonts in the 75dpi directory.

Font databases are created by running the *mkfontdir(n)* program in the directory containing the compiled versions of the fonts (the *.pcf* files). Whenever fonts are added to a directory, *mkfontdir* should be rerun so that the server can find the new fonts. If *mkfontdir* is not run, the server will not be able to find any fonts in the directory.

11.2.4.2 Obtaining Fonts From a Network Font Server

By default, the X server will look for fonts from the local directories described in the previous section. To obtain fonts from a network font server when the X server starts up, follow these steps:

- Copy the file */usr/lib/X11/xinit/xserverrc* to your home directory as *.xserverrc*:

```
cd
cp /usr/lib/X11/xinit/xserverrc .xserverrc
```

- Edit the *.xserverrc* file and find the line:

```
#/usr/bin/X11/X -fp tcp/gordo:7000
```

  Change this line to:

```
/usr/bin/X11/X -fp tcp/hostname:7000
```

  where *hostname* is the name of the network host running a font server program.

11.2.4.3 Providing Fonts Over the Network

The *fs(n)* program is a font server that runs on your Mach^{Ten} system and provides fonts to other network based X terminals and servers.

Fs has many configuration options which are described in detail in the online manual page. In most environments, *fs* will run using the default configuration.

To run the font server, perform the follow steps:

- Make certain you have installed the *Fonts* software component from your X Window Software distribution disks.

  As the *root* user, start the *fs* program:

  ```
  /usr/bin/X11/fs &
  ```

- To start the font server automatically when Mach^{Ten} boots, add an entry to your */etc/rc.local* file:

  ```
  if [ -x /usr/bin/X11/fs ]; then
      ($ECHO "Starting the font server")    >/dev/console
           /usr/bin/X11/fs &
        fi
  ```

  A template */etc/rc.local* file is in the */usr/share/skel* directory.

11.2.5 X Display Management Under Mach^{Ten}

The *xdm(n)* program manages a collection of X displays, which may be on the local host or remote servers. The design of *xdm* was guided by the needs of X terminals as well as the X Consortium standard XDMCP, the X Display Manager Control Protocol. *Xdm* provides services similar to those provided by *init(1), getty(1)* and *login(1)* on character terminals: prompting for login name and password, authenticating the user, and running a *session*. When your Mach^{Ten} system is configured to run the X server with *xdm*, your users will be presented with a friendly, uniform entry into the X Window system.

Xdm has many configuration options which are described in detail in the online *xdm* manual page. This section will concentrate on configuring Mach^{Ten} and the X server to be controlled by *xdm*.

11.2.5.1 The Mach^{Ten} X Server and XDM

On a typical UNIX workstation, the *xdm* program will launch the local X server and present an *xdm login* window on the local display. When the user session completes or the X server quits, *xdm* is notified and restarts the server with a fresh login window.

Because Mach^{Ten} must share the X window environment with the Macintosh window environment, *xdm* is prevented from arbitrarily starting the X server. Instead, *xdm* is typically started at Mach^{Ten} boot time via an entry in the */etc/rc* file. When the X server is subsequently launched with special XDMCP start up options, it communicates with the *xdm* process via the XDMCP protocol and the login window appears. XDMCP server parameters are described in section "11.2.3.1 X Server Start Up Options".

Under Mach^{Ten}, the X server may only run after a user logs in through the Mach^{Ten} Console Window and starts the *wind(8)* program. Thus, the *xdm* login window may only be displayed after entering the Mach^{Ten} Windows environment via a user account in */etc/passwd*. Mach^{Ten} contains features to automate the process of launching the operating system, logging in and creating a default Mach^{Ten} window environment. These features are described in detail in a technical note in the */pub/tech_notes* directory on *ftp.tenon.com* and summarized in the procedure below. Once logged in, you can automate the launching of the X server and its management by *xdm*. The combined features will allow your X server environment to be completely controlled by the X Display Management system from the moment your Macintosh is powered on.

Automatic Launch of an XDM Managed X Server. System managers will typically configure *xdm* to start the X server and display a login window when Mach^{Ten} boots. This can be accomplished by the following configuration steps:

- Add an entry to your */etc/rc* or */etc/rc.local* file to launch *xdm* when Mach^{Ten} boots:

```
if [ -x /usr/bin/X11/xdm ]; then
  ($ECHO "Starting xdm")              >/dev/console
   /usr/bin/X11/xdm
fi
```

- Add an account to your */etc/passwd* file. The account will serve as a default user shell from which actual user *xdm* sessions are started:

```
adduser xdmuser -p MyXdmPass -s /usr/bin/csh
```

- In the *.login* file in the home directory of the *xdmuser* account, add the environment variables XDM_INIT and XDM_RESET just prior to the invocation of the *wind* program:

```
setenv XDM_INIT
setenv WIND_RESET
exec /usr/etc/wind
```

  The XDM_INIT variable will prevent *xinit(n)* from attempting to launch a default client environment when the server starts up. The default environment will be taken instead from a *.xsession* file in the home directory of the user logging in through the *xdm* window.

  The WIND_RESET variable will cause the Mach^{Ten} Window environment to exit when the X server window is closed via the Mach^{Ten} File menu item.

- Create a file named */usr/local/do_console_display* and add the name of the *xdmuser* account:

```
xdmuser
```

- Edit the */etc/ttys* file and find the line:

```
console "/etc/getty std.9600"   vt100   on secure
```

  Change this line to:

```
console "/etc/getty display std.9600"   vt100  on secure
```

  Mach^{Ten} will recognize the **display** variable in this file at boot time and automatically log into the account specified in */usr/local/do_console_display*.

- Create a *.windrc* file in the home directory of the *xdmuser* account with a single entry *x* in the file. This entry will cause the X server to launch automatically when the *xdmuser* account starts up the Mach^{Ten} Windows environment.

- Create a *.xserverrc* file in the home directory of the *xdmuser* account with the *-query* XDMCP server start up option:

```
    #!/bin/sh
    /usr/bin/X11/X -query localhost
```

  When the server starts up, it will communicate with *xdm* to display the *xdm* login window on the local display.

- You can further automate the process to display an *xdm* login window when your Macintosh is powered on by using Finder to place an alias to the Mach^{Ten} application (in the */MachTen* folder) in your System Folder Start Up Items folder.

The XDM Session. When a user logs into the *xdm* login window, the user session is started. The session is controlled by an executable shell script in the logged in user's home directory called *.xsession*. If this file is not found, *xdm* runs a default session from the file */usr/lib/X11/xdm/Xsession*. You can copy your local *.xinitrc* file to *.xsession* and customize it as needed. When the last client (typically the window manager client) in the *.xsession* file exits the session completes and *xdm* presents a new login window.

The *.xsession* file must have *execute* privilege. Use the command:

```
    chmod 755 .xsession
```

to make the *.xsession* file executable.

11.2.6 X Over Serial Lines

Under Mach^{Ten}, you may run X client applications over serial lines to display on your local X server using the Serial Line Internet Protocol (SLIP), the Point-to-Point Protocol (PPP) or, with the aid of a remote AppleTalk to Internet router, Apple Remote Access.

Information on configuring PPP or SLIP for Mach^{Ten} is found in sections "9.13.8 Using PPP" and "9.13.9 Using SLIP", and in the */pub/tech_notes* directory on *ftp.tenon.com*.

The *-wm* backing store X server start up option is a potentially useful option when displaying X clients over slow serial lines. This option causes the server to save a local copy of a window's contents when the window becomes obscured by an overlapping window. As windows are rearranged and the obscured window's contents become exposed, the window is instantly redrawn by the X server. With the option disabled, the remote X client must retransmit the window's contents over the serial line when the window becomes exposed.

Refer to section "11.2.3.1 X Server Start Up Options" for details on how to start the X server with the *-wm* option. The X server uses more system memory when started with the *-wm* option.

11.3 Mach^{Ten} X Window Software Overview

This release of the Mach^{Ten} X Window Software is based on the public release of X11, release 5, patch level 23 from MIT. Its purpose is to provide the necessary tools to allow for the development and display of X applications under Mach^{Ten} on a Macintosh. To that end, the release contains an X Window server, fonts, a font server, programming libraries, configuration files, and documentation from the X11 R5 release adapted for use under Mach^{Ten}. Programming and server environments in the Motif and OpenLook GUI style, based on OSF Motif 1.1.4 and Sun Open Look 3.1, are included. The release also includes a suite of public X client applications in source form as a reference aid in program development.

The main display server components provided are:

XMachTen	The Mach^{Ten} X Window Server
mwm	The Motif Window Manager and user environment
olvwm	The OpenLook Window Manager and user environment
fonts	X11 fonts in Portable Compiled Format and a network font server

The main client development and run-time library components provided are:

X	The X11 library containing the core of the X protocol
Xt	The X Intrinsics toolkit library
Xaw	The Athena Widget set of buttons, pull-down menus, labels, etc.
Xmu	The X Miscellaneous Utility used by the Athena widget set
Xau	The library supporting a security authorization for client/server connections
Xext	The library supporting the Shape, Multi-buffering and MIT-misc extensions
Xi	The library of the Input extension to the protocol supporting alternative input devices
Xm	The Motif widget library
UIL	The Motif User Interface Language compiler
Xview	The Xview widget library
Olgx	The OpenLook graphics library
Phigs	The Application Protocol Interface library supporting the creation of PEX protocol, version 5.1

PEX5	The Xlib-level C interface library to the PEX protocol, version 5.1
Xdmcp	The library of X Display Manager Control Protocol routines
Imake	The configuration system for generating *Makefiles*
Clients	A complete source distribution of MIT client and demonstration programs

The software in this release is contained on CD-ROM. When installed, the X11 software will reside in the following directories after the installation:

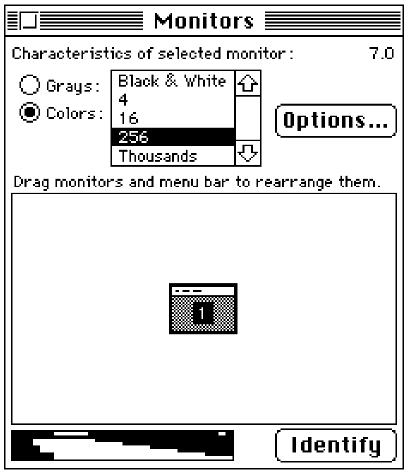
/usr/X11/lib	X11 library archive files (libX*.a)
/usr/lib/X11	X11 executable files
/usr/bin/X11	X11 configuration files, fonts and application default resource files
/usr/include/X11	X11 *include* files for application development
/usr/include/Xm	*include* files for Motif application development
/usr/X11/mann	X11 manual pages
/usr/include/Xview	*include* files for OpenLook application development
/src/X11/R5/mit	Source files for the MIT X client and demonstration programs. Pre-built Mach^{Ten} executables for these programs are found in */usr/bin/X11*.
/src/X11/R5/contrib	Contributed X application source files

11.3.1 Preparing Your Macintosh Control Panels

When launched, the X server will configure itself based on the settings in the *Monitors*, *Sound*, *Mouse*, and *Keyboard* Control Panels. The X server's performance characteristics can be tuned in the *MachTen Controls* control panel.

You must visit the *Monitors* control panel prior to running the X server.

Open your *Monitors* control panel and select from the *Black and White* or *256* Colors or Grays settings. The X server is designed to run using one of these settings.



Characteristics of selected monitor: 7.0

Grays: [✓ Colors:] 256

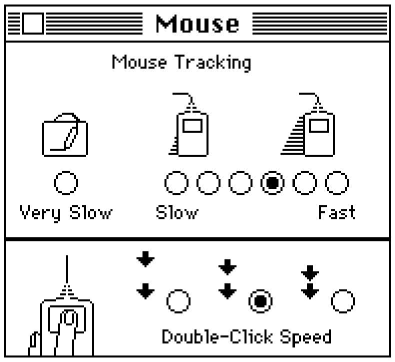
Drag monitors and menu bar to rearrange them.

Figure 44. The Monitors Control Panel

In a multi-screen or video mirroring configuration, the X server runs on the main screen (the screen containing the menu bar).

You are prevented from changing the monitor characteristics while the X server is running. Doing so will cause the X server to exit abruptly.

Open your *Mouse* control panel and review your mouse tracking and double-click rate. The X server obeys the settings in this control panel and ignores requests from X clients to change mouse characteristics.



Mouse

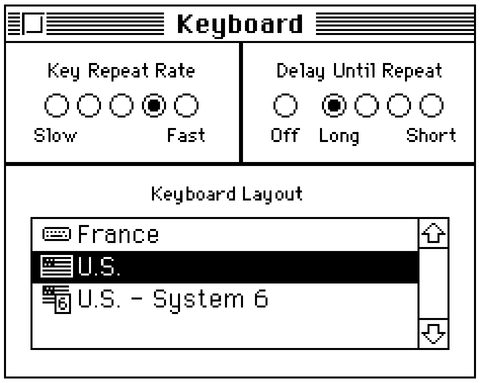
Mouse Tracking

○ Very Slow ○ Slow ○ ○ ● ○ ○ Fast

○ ● ○ Double-Click Speed

Figure 45. The Mouse Control Panel

Open your *Keyboard* control panel and review your keyboard repeat rate and delay, if any. The X server obeys the settings in this control panel and ignores requests from X clients to change keyboard repeat frequency.



Keyboard

Key Repeat Rate: Slow ○ ○ ○ ● ○ Fast

Delay Until Repeat: ○ Off ● Long ○ ○ ○ Short

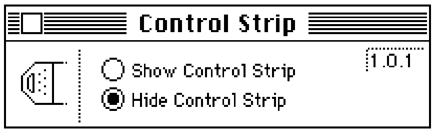
Keyboard Layout

France
[✓ U.S.]
U.S. - System 6

Figure 46. The Keyboard Control Panel

Open your *Sound* control panel and review the sound and intensity. X clients that ring the system bell will generate the selected sound. The X server will also honor X client requests to change bell duration and to turn on or off the bell. These requests will not affect your Sound control panel settings.

If you have a *Control Strip* control panel, select the Hide Control Strip option when running the X server.



Control Strip

1.0.1

Show Control Strip [✓ Hide Control Strip]

Figure 47. The Control Strip Control Panel

Open your *MachTen Controls* control panel. When the Scheduling Priority slide bar is moved upward toward the **Unix** mark, the X server will draw output faster to the screen. When the slide bar is moved downward toward the **Mac** mark, keyboard input and window manipulation (re-positioning, re-sizing, etc.) response will be quicker. The recommended slide bar setting that optimizes these potentially competing demands on the X server is shown below.

Always close the *MachTen Controls* control panel to activate the slide bar settings.

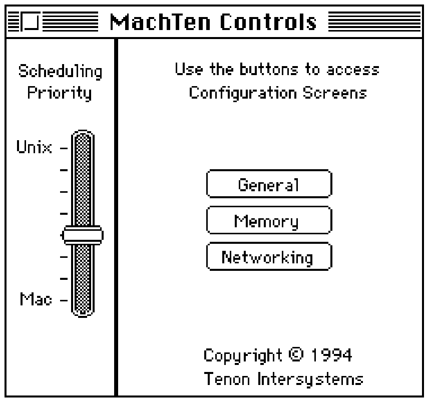

MachTen Controls
Scheduling Priority
Unix
Mac
Use the buttons to access Configuration Screens
General
Memory
Networking
Copyright © 1994 Tenon Intersystems

Figure 48. The Mach^{Ten} Control Panel

11.3.2 Getting Started With X

Following is a road map for information on starting, customizing, operating, and maintaining the X Server.

- Beginning launch and use of the X server is found in section “11.1 The X Desktop”.
- If you want to reconfigure the default X server or learn more about its capabilities before launching it, refer to section “11.2.3 The X Server Program”. Three button mouse, remote font server and International (non-US) keyboard users must consult this section before launching the X server.
- The online manual pages *XMachTen*, *XServer* and *X* are useful and convenient references.
- Some frequently asked questions about the X server, including troubleshooting issues, are answered in section “11.4 X Window Software Frequently Asked Questions”.
- General information on the operation of Mach^{Ten} is found in this User’s Guide.

11.3.3 Building X Applications

All distributed X client executable files in the */usr/X11/bin* directory have been created from the source files in */src/X11/R5*. This section discusses the general procedure for creating X client applications under Mach^{Ten}.

The *Makefile* in X client software is generated automatically by the *imake* program. The program combines machine-independent descriptions (called *Imakefiles*) of targets to be built with machine-dependent sets of parameters. The *xmkmf(n)* script in *usr/bin/X11/* invokes *imake* to build the *Makefile* from an *Imakefile*.

1. To build a single X client, e.g. *xlogo*, make the *Makefile* and executable:

   ```
   cd /src/X11/R5/mit/clients/xlogo
   xmkmf
   make
   ```

2. To install the application(s) in */usr/bin/X11* type:

   ```
   make install
   ```

3. To install the application manual page(s) in */usr/X11/man/mann* type:

   ```
   make install.man
   ```

4. If you chose to install the client(s), you may free up disk space by deleting the extra copy of the executable image(s):

   ```
   make clean
   ```

To build the entire X client suite, follow these additional steps:

5. Make the master *Makefile* using the *xmkmf* utility:

   ```
   cd /src/X11/R5/mit/clients
   /usr/bin/X11/xmkmf
   ```

6. Build the application *Makefiles* and executables:

   ```
   make Makefiles
   make
   ```

   then follow steps 2 through 4, above.

Compiler warning messages are expected from some clients in the MIT distribution. These warnings will not adversely affect the client’s operation under Mach^{Ten}.

11.3.3.1 Running X Client Applications

The build process may be tested by executing an application program compiled and installed in the previous step and instructing it to display on the local X server. Online manual pages are available for each of the X client applications.

Make sure the X server is running (refer to the previous section). Start the client:

```
    /usr/bin/X11/xlogo &
```

You can also display the client on a remote X server on your network. You should verify that your networking facilities are working properly before trying to display an X client on a remote server. If you cannot connect with the server via *telnet* and *rlogin*, X applications will probably fail as well.

The X server display name is given on the command line after the *-display* argument in the form `host:display`. In the example below, the *xlogo* client is displayed on display number *0* of X server host *robin*.

Make sure */usr/bin/X11* is in your directory search path.

```
    xlogo -display robin:0 &
```

You may also change the DISPLAY variable in your environment. If *-display* is not found on the command line, X applications connect to the server indicated by the DISPLAY environment variable. Under Mach^{Ten}, the default DISPLAY (as set in *.login* or *.profile* in your home directory) is the local host (*unix:0* or simply *:0*).

Under csh (*.login*):

```
    setenv DISPLAY robin:0
```

Under sh (*.profile*):

```
    DISPLAY=robin:0
    export DISPLAY

    xlogo &
```

11.3.3.2 The X11 Application Development Environment under Mach^{Ten}

The *imake* configuration utility is included to generate machine-specific *Makefiles* from machine-independent *Imakefiles*. Another utility, called *makedepend*, is provided to generate *Makefile* dependencies for C language files. The *xmkmf* shell script in *usr/bin/X11/* is used to create *Makefiles* from *Imakefiles*. The easiest way to construct an *Imakefile* is to start with one that does something similar and modify it. The various macros that are used in an *Imakefile* are defined in the file */usr/lib/X11/config/Imake.rules*. Examples of *Imakefiles* can be found under the */src/X11/R5/mit/clients* directory. You are strongly urged to use *imake* and *makedepend* so that your software will work across releases.

The configuration files for *imake* are located in the directory *usr/lib/X11/config/*. *Makefiles* are created from a template file named *Imake.tmpl*, a machine-specific *.cf* file, and a site-specific *site.def* file. The template file should not be modified.

The file *Tenon.cf* defines the specific configuration for application development under Mach^{Ten} and need not be modified.

The following suggestions are offered by the MIT X Consortium on writing X applications that are portable to other hardware platforms:

1. Keep all source file names to 12 characters or less. This is the maximum number of characters that older System V file systems allow when using a source code control system.

2. If you absolutely must use *Makefiles* instead of *Imakefiles*, link against -lX11 instead of -lX. If you are using *imake*, use the symbolic names $(XAWLIB), $(XMULIB), $(XTOOLLIB), $(EXTENSIONLIB), and $(XLIB). *Xaw* clients may use the symbol *XawClientLibs* to refer to the appropriate libraries.

3. Include header files using the syntax *<X11/file.h>* instead of *X11/file.h*, *<X/file.h>*, or *X/file.h*.

4. Include *<X11/Xos.h>* if you need *types.h*, *string.h* or *strings.h* (then use the routines *index* and *rindex* instead of *strchr* and *strrchr*), *file.h*, *time.h*, or *unistd.h*.

5. If you need to put in System V vs. BSD dependencies, use *#ifdef SYSV*. If you need SVR3 vs. SVR2, use *#ifdef USG*.

6. Do not assume that the root window's Visual (returned by the *DefaultVisual* macro) is the only one available. Some color screens may use a black and white window for the root or could provide *StaticColor* as well as *PseudoColor* visuals. Unfortunately, most libraries do not have adequate support for locating visuals to use. In the meantime, use *XGetVisualInfo()*.

7. Use *-display displayname* to specify the X server to contact. Do not simply assume that a command line argument that has a colon in it is a *displayname*. If you accept command line abbreviations, make sure that you also accept the full *-display*.

8. Use *-geometry geomspec* to specify window geometry. Do not simply assume that command line argument that begins with an equal sign is a window geometry. If you accept command line abbreviations, make sure that you also accept the full *-geometry*.

9. Use the *.man* suffix for program manual page sources.

11.3.3.3 Programming Notes

- Mach^{Ten} supplies each UNIX process with a fixed 36K byte stack when running in non-virtual memory mode (the *unix* kernel). Applications that define large stack-based storage should add the following lines to the associated *Imakefile* requesting a larger process stack when the program is linked:

  ```
  #ifdef MachTenPPCArchitecture
  LOCAL_LDFLAGS = -Xlstack=<stacksize>
  #endif
  ```

  To increase the application stack size by 100K bytes, use the command:

  ```
  LOCAL_LDFLAGS = -Xlstack=100000
  ```

- Refer to the *PortingProblems* technical note in */pub/tech_notes* on *ftp.tenon.com* for further information on programming under Mach^{Ten}.

11.4 X Window Software Frequently Asked Questions

Some frequently asked questions about the X Window Software are answered below:

- **The X server does not start up or dies unexpectedly. What is wrong?**

  You must be running *wind(8)* when you start the X server. Log into an account under Mach^{Ten} and be certain the *wind* process is running before launching the X server.

  The X server depends on */usr/bin/X11* present in your directory search path. Under Mach^{Ten} for the PowerPC, */usr/bin/X11* is soft linked to */usr/X11/bin*. *XMachTen*, the X server binary, and all server support applications reside in this directory. If you are running a custom shell (e.g., *tcsh* or *ksh*), make certain your X11 search path is set up **before** you start the *wind* process. The sample *.login* and *.profile* files in */usr/share/skel* show examples of adding to your search path to include */usr/bin/X11*.

  You may have accidently installed the X software on a target volume other than the volume containing your Mach^{Ten} folders.

  You may have moved your Mach^{Ten} folders to a subfolder on the target volume prior to installing the X Window Software. Use Finder to temporarily relocate all Mach^{Ten} folders to the top level of the target volume before installing the X software.

  Read section "11.3.1 Preparing Your Macintosh Control Panels".

  The X server maintains a log file in */var/adm/X0msgs*. Consult this file if you have difficulties starting or operating the X server. You can redirect entries logged to this file to the Mach^{Ten} Console terminal window using the command:

  ```
  ln /dev/console /var/adm/X0msgs
  ```

- **My arrow keys stop working when I run the X server. Why?**

  By default, the X server uses the left-arrow for mouse button two, the right-arrow for mouse button three, the down arrow as a *Control* key, and the up arrow as a *Meta* key. When the *Option* key is pressed with an arrow key, the arrow key's original function is temporarily restored. When you re-map mouse button keys two and three in your *.xserverrc* file, all arrow keys' original functions are permanently restored. You may also use the *-mo* server start up option to restore normal functionality to your arrow keys.

  Refer to section "11.2.3.2 Mouse Button Mapping" and the example in the sample X server start up file */usr/lib/X11/xinit/xserverrc*.

- **What three button mouse do you recommend?**

  The Logitech MouseMan has been tested successfully with the X server.

- **My X desktop Macintosh menu bar changes colors when I run certain X color applications. Why?**

  When the X desktop is visible, X client applications have precedence over the MacOS when competing for the limited number of colors that can be displayed simultaneously. The menu bar is drawn by the MacOS based on the best matching colors available.

  Consider running the X server with no Macintosh menu bar as described in section "11.2.3.1 X Server Start Up Options".

- **My disk is full. What X software can I prune from my disk without affecting X server operation?**

  You will recover approximately 11 MB of disk space by removing the X fonts from */usr/lib/X11/fonts* directory. You must obtain fonts for the X server from a network font server if you remove the local fonts. Refer to section "11.2.4 Font Service".

  The X client source code in the */src/X11* directory is not necessary to run the X server. Deleting the source will recover 7 MB of disk space. If you keep the source tree after building and installing the X clients, run *Make clean* from the */src/X11/R5/mit/clients* directory to remove object and temporary binary files.

  Removal of the X client development libraries from */usr/lib/libX** and the *X11*, *Xm*, *Mrm*, *UIL*, *olgx* and *xview* include directories from */usr/include* will save over 11 MB of disk space. If you are not actively developing X client applications, you can remove these libraries and include files from your system.

  The font, source, and X client development components can be individually installed as needed using the Mach^{Ten} X Window Software Custom Installer.

- **When I type *twm*, Mach^{Ten} responds *twm: Command not found*. Where are the clients located on my system?**

  Make certain the */usr/bin/X11* directory is in your directory search path.

- **I am using a French keyboard. When I type an 'a' into an X window application, the application echoes a 'q'. Why?**

  You must load a French keyboard mapping file when the X server is launched to re-map the default U.S. X keyboard map. Refer to section "Keyboard Mapping".

- **I have a *.Xdefaults* file in my home directory, but my X applications are not affected by its contents. Why?**

  The X server has a local database which takes precedence over the *.Xdefaults* file. This database originates from a previous running of the *xrdb(n)* program with the *-load* or *-merge* options.

  You can force your *.Xdefaults* file to be read by the application by defining the XENVIRONMENT shell variable:

  ```
  setenv XENVIRONMENT .Xdefaults
  ```

  Resources defined via XENVIRONMENT override duplicate resources contained in the X server's database.

  Alternatively, the *.Xdefaults* file is read if the database is removed from the server:

  ```
  xrdb -remove
  ```

- **The delete key does not operate when I run Motif applications. Why?**

  You must supply a key translation resource for applications that use the Motif text widget. A sample resource file containing the correct translation is in */usr/lib/X11/xinit/Xresources*. Merge the resource into the X server database using *xrdb* or add it to your *.Xdefaults* file (see above).

- **My X application requests a large color cursor. Why does it show up on the server as a small arrow?**

  The X server takes advantage of the Macintosh firmware cursor for maximum performance and a lively tactile response. The Macintosh cursor is limited to a fixed 16 x 16 pixel size, so when an application tries to display a cursor larger than 16 x 16 pixels, the server substitutes a standard arrow cursor.

  The X server currently does not support color cursors.

- **How do I use different colors for the application I am running?**

  If you have a color monitor, you can specify a custom foreground and background color for your application using the *-fg* and *-bg* command line options. For example:

  ```
  xlogo -bg yellow -fg green &
  ```

  Alternatively, you can copy the default application resource file (for example, *XLogo*) from */usr/lib/X11/app-defaults* (if present) for the program you are running to your home directory and modify it to suit your tastes:

  ```
  XLogo*background: green
  XLogo*foreground: yellow
  ```

  Refer to the manual page for the client you are running for a description of the color resources available to the application.

  Note that command line options take precedence over resource definitions.

- **Where can I obtain other X client applications for Mach^{Ten}?**

  A directory of contributed (and unsupported) X client applications in Mach^{Ten} executable format is maintained on Internet host *ftp.tenon.com* in the */pub/applications/X11* directory.

- **Why does my X application crash when I am using the Mach^{Ten} *unix* kernel?**

  The application has a fixed 36KB stack size when operating in a real memory environment (*unix*) and may require more stack space in order to run. Increase the program's stack allocation using the example in section "11.3.3.3 Programming Notes".

- **I use *emacs*. I noticed that *Meta-%* and *Meta-!* don't do what I expect them to do. What is the problem?**

  The MacOS reserves all *Command-Shift-number* keystroke combinations for special effects. Since the default X Meta key is the Command key, the *Meta-Shift-number* key sequence is not delivered to the X client. Use the *xmodmap(n)* utility to re-map Meta to something other than the Command key.

11.5 References & Suggested Reading

- OSF/Motif User's Guide, Open Software Foundation

The following books published by O'Reilly & Associates are recommended:

- Volume 1: Xlib Programming Manual

- Volume 2: Xlib Reference Manual

- Volume 3: X Window System User's Guide
  Details the X Window System and common X client applications.
  This volume is available in an OSF/Motif Edition focusing on the Motif user environment and *mwm*.

- Volume 4: X Toolkit Intrinsics Programming Manual

- Volume 5: X Toolkit Intrinsics Reference Manual
  Comprehensive guides to programming with the X library and X Toolkit.

- Volume 6: Motif Programming Manual
  Programmer's guide to creating Motif applications using the Motif toolkit.

- Volume 8: X Window System Administrators Guide
  Comprehensive guide to administering the X Window System. This volume is a highly recommended companion to the Mach^{Ten} X Window Software package.

The following Nutshell Handbooks from O'Reilly & Associates my also be helpful:

- Managing Projects with make
- Managing NFS and NIS
- The Whole Internet User's Guide
- Practical Unix Security
- Essential System Administration
  TCP/IP Network Administration

INDEX

A
access control,
  host based, 187
  using XDMCP, 191
adduser, 50
administration, 43
app-defaults files,
  See resources
Apple file sharing,
  effect on X performance, 188
AppleDouble, 90
AppleSingle Encapsulation, 72
AppleTalk, 25
archiving, 76
arrow keys,
  default mapping, 184, 203
  remapping using mouse button options, 182, 184

B
background program execution, 57
backing store,
  -*bs* server start up option, 181
  -*wm* server start up option, 183
  use with X clients over serial lines, 193
backing up, 76
bash, 11
BBEdit, 5, 75
Berkeley r-series, 114
bwnfsd, 86

C
C++, 146
characters,
  Table of Special Keyboard Characters, 56
chcreat, 73
chtype, 73
client mounting, 81
clients,
  See X client applications
code fragments, 140
color,
  Monitors control panel setting of, 196
  problems using, 204, 206
  resource specifications, 206
color (continued),
  specifying on the command line, 169, 206
Command key,
  use as X Meta key, 184, 187
CommToolBox devices, 133-134
communications, 115
  baud rates, 116
  cabling, 115, 117
  connectors, 115
  GPI serial cable, 117
  modem connection, 117
  serial ports, 115, 117
  terminal connection, 117, 118
  using PPP, 118
  using SLIP, 123
configuration screens, 22-27
  general, 22
  memory, 23
  networking, 24
    example, 28
control panels,
  Control Strip, 198
  Keyboard, 197
  Mach^{Ten}, 21, 199
  Monitors, 196
  Mouse, 197
Control Strip control panel,
  configuration, 198
cross development tools, 143

D
daemons,
  Table of Mach^{Ten} Daemons, 31
data files,
  Table of Mach^{Ten} Data Files, 32
debugging, 151
default gateway, 27
development tools, 4
device, 61
dfork, 73
dfork.text, 74
directory tree,
  usr, 62
  root, 61
  var, 62

disk cache, 12
disk space,
    minimizing utilization of, 200, 204
DISPLAY environment variable,
    in *.login* and *.profile* files, 188
    use in running X applications, 201
display management,
    See also XDMCP
    under Mach^{Ten}, 191-193
    session file, 193
domain name, 27
    server, 27, 106
dynamic
    linking, 3
    memory configuration, 3

E
emacs, 11
    meta keys, 206
errors,
    server logging of, 188
Ethernet, 26

F
file maintenance, 75
file system,
    Mach^{Ten}, 61
    Macintosh file manager, 61
    structure, 61
    UNIX file system, 61
    hierarchy for software development, 142-143
files,
    accessing directly from CD-ROM, 18-19
    accessing Macintosh files from UNIX applications, 72
    AppleSingle Encapsulation, 72
    differentiating Macintosh and UNIX, 72
    group file, 53
    linked, 65
    locked, 66
    major system administration files, 63
    naming, 64
    password file, 52
    permissions, 69
    utilities for manipulating Macintosh files, 73
floating root, 62
floppies,
    formatting, 72
        raw, 72
fonts,
    obtaining from remote font server, 190
        -fp server start up option, 182
    server default font path, 189
font server,
    configuration under Mach^{Ten}, 190
Fortran, 146
FPU,
    See floating point hardware
frequently asked questions, 203
FTP, 113

G
Gcc, 145
GNU info files, 6, 11

H
header files, 145
heap size,
    setting, 140
help,
    getting started with X, 199
home directory, 48
host name, 22

I
imake,
    use of under Mach^{Ten}, 201
incoming connections, 25
incoming mail, 25
installation, 5
    See also X client applications
    launching Mach^{Ten}, 9
    on hard disk, 8
    optional components, 11
    overview, 194
    site preparation, 6
international keyboard keys, 186
internet, 105
internet protocol,
    IP addresses, 24-25
    IP forwarding, 25
IP,
    addressing strategy, 122
    forwarding, 111

K
kernel, 2
    parameters, 23, 24
Keyboard control panel,
    configuration, 197
  keyboard mapping, 186
    default arrow key mapping, 184, 204
    international keyboard layouts, 186
    using keys as mouse buttons, 184
killing a program, 57

L
launching Mach^{Ten}, 29
libraries, 140-142
    naming conventions, 141
    shared, 140
    traditional UNIX, 142
loader, 146
login,
    accounts, 48
    administration, 54
    console, 32-33
    directory, 48
lwsrv, 98
    configuration files, 99
    enabling, 100

M
Mach^{Ten},
    Network File System (NFS), 79
    windows, 34
        designing a special environment, 46
    Windows menu,
        Apple menu, 35
        Edit menu, 36
        File menu, 36
        Fonts menu, 41
        Order sub-menu, 39
        Positions sub-menu, 38
        Size sub-menu, 38
        Window menu, 37
Mach^{Ten} Controls control panel,
    configuration, 199
    scheduling priority slide bar,
        effect on X server performance, 199
Macintosh,
    applications,
        creating, 149
    OS interface libraries, 149
Macintosh Debugger, 153
MacsBug, 152
mactcp, 111
mactext, 73
makefiles, 145
man pages, 5, 43
maximum transmission unit (MTU), 27
    setting, 27, 127
    SLIP MTU, 27
    TokenRing MTU, 27
memory,
    mapped file access, 4
    allocations of, 139
    calculating requirements, 149
    real, 147
menu bar, 159-161
    functions,
        file menu, 159
        order sub-menu, 161
        quitting the X server, 162
        window menu, 160
        hiding, 161
            via *-fs* server start up option, 182
            via *-menu* server start up options, 182
        shortcuts,
            See xtmenu program
message of the day, 55
meta keys, 187
    use of in *emacs*, 206
Metrowerks Debugger, 153
mkpef, 146
Monitors control panel,
    configuration, 196
Motif,
    See also window managers
    environment, 170
mount, 81
mount(8), 71
mounting,
    Macintosh volumes, 70
    permissions, 71
    automatic of removable media, 71

mouse,
  three button configuration, 184-185
  use in window manager, 171
Mouse control panel,
  configuration, 197
multiple screens, 196
mwm,
  See window manager

N
Netmasks, 27
networking, 103
  control panel, 106
  domain name resolver, 108
  electronic mail, 111
  importing Macintosh image files, 109
  IP forwarding, 111
  Mach^{Ten} as a router, 110
  Mach^{Ten} as domain name server, 106
  Mach^{Ten} as internet host, 105
  Mach^{Ten} as POP server, 111
  Mach^{Ten} as Web server, 108
  multihoming Web server, 109
  multiple interfaces, 106
  table of Mach^{Ten} TCP/IP protocol stack, 103
  using Berkeley r-series, 114
  using FTP, 113
  using NFS, 79, 114
  using Telnet, 114
NFS, 79
  access privilege strategies, 90
  AppleShare access privileges, 89
  desktop NFS, 84
    authentication options, 86-87
    DOS extensions, 91
    folders,
      bulletin board, 90
      drop, 90
      locked, 90
      shared, 90
    record locking, 85
    volume mounting, 85
  file system security, 88
  mapping AppleShare access privileges, 89
  setting up, 80, 83
  UNIX access privileges, 88

O
olvwm,
  See window manager
OpenLook window manager,
  See olvwm
optional installations, 11

P
passwd, 52
PEF, 3, 140
performance,
  tuning, 11
    system optimization, 12-14
  tuning the X server for, 188, 198
perl5, 11
POP server, 111
ported applications, 6, 11
ported X applications, 188, 205
powerPC executable format (PEF), 3, 140
PPP, 118
  configuring Mach^{Ten} system as client, 121
  use with X over serial lines, 193
print spooler database, 93
print spooling, 93
printing, 93
  ImageWriter, 94
  LaserWriter, 94
  local, 94
  remote, 96, 98
  notification of status, 101-102
  receiving remote printing, 97
  selecting an alternate printer, 97
priority slide bar, scheduling, 21
programming, 137

Q
quitting Mach^{Ten}, 47

R
recommended reading, 207
reconfiguration,
  automatic, 16-17
  manual, 17-18
reinstallation, 15-16

resources, 166-169
  in *.Xresources* file, 168
  app-defaults files, 168
  color specifications, 206
  comparison to Macintosh application
    preferences, 166
  location in file system, 167
  precedence for use, 167-169
  recommendations for use, 169
  syntax, 166
  XENVIRONMENT variable, 168
rfork, 73
root password, 49
route, 110
router, 110

S
screen savers,
  Macintosh, 188
  xlock, 188
search path, 12
security, 49
server,
  See X server
  exporting, 81
shell files, 58
shells, 32
SLIP, 123
  configuring Mach^{Ten} a a dial-in server, 127-129
  starting, 130
  use with X over serial lines, 193
software development tools, 137-138
Sound control panel,
  configuration, 198
startup,
  automatic, 45
  environment, 44
  manual, 46
superuser, 49
symbol information, 147
system limits, 15
system.mwmrc file,
  See window managers, Motif

T
TCP,
  comparison with MacTCP, 103
  protocol stack in Mach^{Ten}, 7
telnet, 114
Tenon.cf configuration file, 202
text,
  combining Macintosh and UNIX, 73
  copy and paste, 75
  editing, 74
  files, 73
The Debugger, 153
three button mouse,
  configuration, 184-185
  example using Logitech mouse, 185
  -mbm, *-mbr*, *-mo* server start up
    options, 182-183
time,
  stamps, 69-70
  zone, 23
tip, 123-125
TokenRing, 26
troubleshooting, 19-20
  building and running clients, 200-202, 205,
    206
  color related problems, 204, 205, 206
  server start up problems, 203
  server error logging, 188

U
Unix<->Text, 5, 74
UNIX
  designing a special environment, 46-47
  environment,
    management of, 57
  shell, 48
unix kernel,
  differences with *vmunix* kernel, 206
unmounting,
  automatic, 71
  Macintosh volumes, 70-71
user,
  accounts,
    setting up, 50-51
  logins,
    disabling, 55
users,
  removing, 55

V

video mirroring, 196
virtual memory,
  advantages, 206
vmunix kernel,
  See virtual memory
volume, 61

W

wall(1), 55
web server, 109
window manager,
  general introduction, 170
  hot keys, 163
  Motif,
    icons, 175-177
    menus, 177-179
    moving and resizing windows, 172-175, 179-180
    pointer shapes, 171
    selecting windows, 171
    starting, 170
    quitting, 180
  olvwm, 170
.*windrc* file, 161
writing to all users, 55

X

X client applications,
  building, 200, 201-202
  obtaining contributed clients for Mach^{Ten}, 206
  organization in software distribution, 194-196
  running, 201,
    in the .*xinitrc* start up file, 164-165
    in the *xdm* .*xsession* file, 193
X Desktop,
  defined, 158
  removal, 162
X server,
  access control,
    host based, 187
    using XDMCP, 191-192
  default client start up script, 164-165
  error logging, 188
  launching,
    from .*windrc* file, 161
X server (continued),
  using hot key sequence, 158
    via *xtmenu* program, 162
  quitting, 162
    via *xtmenu* program, 162
  start up options, 182-183
    use in .*xserverrc* file, 181
  tuning for high performance, 188-189, 198
  use of an alternate display number, 188
X Windows, 5
XCOFF, 140
.*Xdefaults* file, 168, 205
xdm,
  See display management
XDMCP,
  See also display management server start up options, 183
XENVIRONMENT variable, 169
xinit program, 164
.*xinitrc* file,
  discussion, 164-165
  use in starting *mwm*, 170
XMachTen program, 163
  See also X server
xmkmf, 200
  See also imake
xmodmap program,
  use with international keyboards, 186
xrdb program,
  precedence over .*Xdefaults* file, 168, 205
  use in loading server resource database, 168
.*Xresources* file, 168
.*xserverrc* file,
  introduced, 189
  use with network font server, 190
  use with three button mouse configuration, 186
.*xsession* file,
  use with *xdm*, 191-192
xterm program,
  default start up client, 158, 166, 170
  example use under *mwm*, 179
xtmenu program, 162, 182