

# Power Mach<sup>Ten</sup>

---

## User's Guide



*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

info@tenon.com • <http://www.tenon.com>

Copyright 1998 Tenon Intersystems  
All Rights Reserved  
Printed in USA

Tenon, Mach<sup>Ten</sup>, Power Mach<sup>Ten</sup> and the Tenon logo are trademarks of Tenon Intersystems.  
X Window System and X11 are registered trademarks of MIT.  
Apple, Macintosh and Power Macintosh are registered trademarks, and Finder is a  
trademark of Apple Computer, Inc.  
UNIX is a registered trademark in the United States and other countries, licensed  
exclusively through X/Open Company limited.  
All other product names are trademarks of their respective holders.

---

# TABLE OF CONTENTS

---

<b>1.0</b>	<b>Power Mach<sup>Ten</sup> — UNIX on the Macintosh Desktop</b>	<b>1</b>
1.1	The Mach <sup>Ten</sup> Desktop	3
1.1.1	Mach <sup>Ten</sup> Terminal Window Desktop	4
1.2	The Mach <sup>Ten</sup> “UNIX Virtual Machine”	6
1.3	The Mach <sup>Ten</sup> Architecture	7
1.3.1	Dynamic Memory Configuration	7
1.3.2	Dynamically Linked, Shared Libraries	7
1.3.3	Memory Mapped File Access	9
1.3.4	Integrated Software Development Tools	9
1.3.5	Native Fast File System	9
<b>2.0</b>	<b>Installing Mach<sup>Ten</sup></b>	<b>11</b>
2.1	Performance Tuning — Optimizing Mach <sup>Ten</sup>	21
2.1.1	System Optimization Guidelines	22
2.1.2	How Can You Tell When a System is Approaching Its Limits?	25
2.2	Reinstallation	26
2.2.1	Reconfiguration	28
2.2.1.1	Automated Reconfiguration	28
2.2.1.2	Manual Reconfiguration	29
2.2.2	Accessing Mach <sup>Ten</sup> Sources from the CD-ROM	30
2.3	Troubleshooting	31

<b>3.0</b>	<b>The Mach<sup>Ten</sup> Control Panel</b>	<b>33</b>
3.1	Scheduling Priority Slide Bar	34
3.2	Configuration Screens	34
3.2.1	General Configuration Screen	35
3.2.1.1	Host Name	35
3.2.1.2	Time Zone	36
3.2.2	Memory Configuration Screen	36
3.2.2.1	Maximum Number Parameters	37
3.2.3	Networking Configuration Screen	38
3.2.3.1	Enable IP Forwarding	39
3.2.3.2	Enable Incoming Mail	39
3.2.3.3	Enable Incoming Connections	39
3.2.3.4	Internet (IP) Addresses	40
3.2.3.5	Netmasks	42
3.2.3.6	Maximum Transmission Unit Settings	42
<b>4.0</b>	<b>Launching Mach<sup>Ten</sup></b>	<b>45</b>
4.1	The Mach <sup>Ten</sup> Login Console	49
4.2	Mach <sup>Ten</sup> Windows	52
4.2.1	The Apple Menu	53
4.2.2	The File Menu	53
4.2.3	The Edit Menu	54
4.2.4	The Window Menu	56
4.2.4.1	The Positions Sub-Menu	57
4.2.4.2	The Size Sub-Menu	58
4.2.4.3	The Order Sub-Menu	59
4.2.5	The Fonts Menu	61

---

<b>5.0</b>	<b>Mach<sup>Ten</sup> Administration</b>	<b>63</b>
5.1	A Word About Man Pages	64
5.2	Tailoring the Startup Environment	65
5.2.1	Setting Up Mach <sup>Ten</sup> to Boot Automatically	67
5.2.2	Manual Startup	67
5.3	Tailoring Mach <sup>Ten</sup> Windows	68
5.4	Tailoring Your UNIX Environment	69
5.5	Quitting Mach <sup>Ten</sup>	70
5.6	Login Accounts	71
5.6.1	The Concept of a Home Directory and a User Environment	71
5.6.2	The UNIX Shell	72
5.6.3	Superuser and Privileges	72
5.6.4	A Word About Security	73
5.6.5	The Root Password	74
5.6.6	Setting Up User Accounts	75
5.6.7	The Password File	77
5.6.8	The Group File	78
5.6.9	Administrative Login Accounts	80
5.6.10	Writing to All Users	81
5.6.11	Disabling User Logins	82
5.6.12	Removing Users	82
5.6.13	Changing the Message-of-the-Day	82
5.6.14	Special Characters	83
5.7	Managing Your UNIX Environment	84
5.7.1	What's Running?	84
5.7.2	Killing a Program	85
5.7.3	Background Program Execution	85
5.7.4	Shell Files	86

---

<b>6.0</b>	<b>The Mach<sup>Ten</sup> File Systems</b>	<b>87</b>
6.0.1	Fast File System (FFS)	87
6.0.2	UNIX File System (UFS)	87
6.1	UNIX Fast File System Overview	88
6.1.1	File System Organization	88
6.1.2	File Names	89
6.1.3	Access Permissions	89
6.1.4	Time Stamps	89
6.1.5	Link Counts	90
6.1.6	Hard Links	90
6.1.7	Symbolic Links	90
6.2	Macintosh Hierarchical File System Overview	91
6.2.1	File System Organization	91
6.2.1.1	Volumes and Folders	91
6.2.1.2	File Contents	92
6.2.2	File Names	92
6.2.3	Access Permissions	92
6.2.4	Time Stamps	93
6.2.5	Aliases	93
6.3	Mach <sup>Ten</sup> FFS	93
6.3.1	FFS Within a File	93
6.4	Mach <sup>Ten</sup> UFS	94
6.4.1	File Names	94
6.4.1.1	Maximum Number of Characters	94
6.4.1.2	Case-Sensitive File Names	94
6.4.1.3	Component Separators	95
6.4.1.4	Non-Printable Characters	95

---

6.4.2	Linked Files	96
	6.4.2.1 Hard Links	96
6.4.3	Directory Link Counts	97
6.4.4	Locked Files	97
6.4.5	File Types	97
6.4.6	File Permissions	101
6.4.7	Time Stamps	102
6.5	Mach <sup>Ten</sup> Root File System Layout	103
	6.5.1 The root Directory Tree	103
	6.5.2 The usr Directory Tree	104
	6.5.3 The var Directory Tree	105
	6.5.4 Major System Administration Files	106
6.6	Mounting Macintosh Volumes	107
	6.6.1 Mounting Permissions	107
	6.6.2 Automatic Mounting of Removable Media	108
	6.6.3 Unmounting Macintosh Volumes	108
	6.6.4 Automatic Unmounting	109
	6.6.5 Formatting Floppies	109
6.7	Accessing Macintosh Files from UNIX Applications	110
	6.7.1 AppleSingle Encapsulation	110
	6.7.2 Differentiating UNIX and Macintosh Files	110
	6.7.3 Utilities for Manipulating Macintosh Files	111
	6.7.3.1 dfork and rfork	111
	6.7.3.2 finderinfo	111
	6.7.3.3 restool	111
6.8	Text File Manipulation	112
	6.8.1 Alternating Between Macintosh and UNIX Text	112
	6.8.1.1 mactext	112
	6.8.1.2 unixtext	113

	6.8.1.3	Unix <-> Text	113
	6.8.1.4	dfork.text	113
6.8.2		Editing Tools	114
	6.8.2.1	UNIX Editors	114
	6.8.2.2	Macintosh Editing Applications	114
6.8.3		UNIX to Macintosh "Copy-and-Paste"	114
6.9		File System Administration	115
6.9.1		Creating File Systems	115
	6.9.1.1	Creating an FFS Within a File	115
6.9.2		Checking/Repairing File Systems	116
	6.9.2.1	fsck	116
	6.9.2.2	Fast File First Aid	117
6.9.3		Mounting File Systems	117
	6.9.3.1	Mounting an FFS Within a File	117
	6.9.3.2	Mounting a Macintosh Volume	118
6.9.4		Unmounting File Systems	118
6.9.5		Removing File Systems	118
	6.9.5.1	Removing an FFS Within a File	118
6.9.6		Space Management	119
6.9.7		Backing Up and Archiving File Systems	120
	6.9.7.1	Tape Devices	120
	6.9.7.2	tar	121
	6.9.7.3	dump and restore	122



---

<b>7.0</b>	<b>The Mach<sup>Ten</sup> Network File System (NFS)</b>	<b>123</b>
7.1	Using NFS	124
7.2	How NFS Works	124
7.3	Setting Up an NFS Server or an NFS Client	125
7.3.1	Server Exporting	126
7.3.2	Client Mounting	126
7.3.3	Set-Up Summary	128
7.4	NFS Volumes on the Macintosh Desktop (Desktop NFS)	130
7.4.1	Desktop NFS Volume Mounting	131
7.4.1.1	Automatic Mounting of Desktop NFS Volumes	132
7.4.2	Macintosh Record Locking on Desktop NFS Volumes	132
7.4.2.1	Bwnfsd UID and GID Mapping	133
7.4.3	Authentication Options	133
7.4.3.1	Credentials of Current User	133
7.4.3.2	Prompt for Username and Password	134
7.4.3.3	Username and Password on Command Line	134
7.4.3.4	User ID and Password on Command Line	135
7.4.4	File System Security	135
7.4.4.1	AppleShare Access Privileges	135
7.4.4.2	UNIX Access Privileges	136
7.4.4.3	Mapping AppleShare Access Privileges into UNIX Access Privileges	137
7.4.4.4	Access Privilege Strategies	138
7.4.4.5	Differences Between Apple File Sharing and Desktop NFS	139
7.4.5	AppleDouble	139
7.4.6	DOS Extensions	140

---

<b>8.0</b>	<b>Configuring NIS Under Mach<sup>Ten</sup></b>	<b>143</b>
8.1	NIS Basic Configuration Steps	143
8.2	Set YP Configuration in /etc/yp.conf	144
8.3	Set NIS Configuration in /etc/nis.conf	144
8.4	Set NIS Database Search Order	145
8.4.1	The Default /etc/nsswitch.conf File	145
8.5	Testing the Basic NIS Configuration	147
8.6	Configure Mach <sup>Ten</sup> for Automatic Portmap and Ypbind Startup	148
<b>9.0</b>	<b>Printing</b>	<b>149</b>
9.1	Mach <sup>Ten</sup> Print Spooling	149
9.1.1	The Print Spooler Database	150
9.2	Local Printing	150
9.2.1	Printing Text Files to a PostScript LaserWriter on AppleTalk	151
9.2.2	Printing to an ImageWriter or a DeskWriter	151
9.3	Remote Printing	154
9.3.1	Spooled Printing to a Remote UNIX Printer	154
9.3.2	Receiving Remote Print Jobs	155
9.4	Selecting an Alternate Printer	156
9.5	Extending Remote Printing to Macintosh Applications	157
9.5.1	Lwsrv Configuration Files	158
9.5.1.1	Font Lists	159
9.5.1.2	PostScript Macros	159
9.5.2	Enabling lwsrv	160
9.6	Print Job Status Mail Notification	161
9.7	Status and lp Management Programs	162

---

<b>10.0</b>	<b>Networking with Mach<sup>Ten</sup></b>	<b>163</b>
10.1	Using OpenTransport or Replacing OpenTransport	164
10.2	Configuring Tenon TCP or Dual Stack Networking	167
10.3	Using Mach <sup>Ten</sup> as an Internet Host	168
10.3.1	The Mach <sup>Ten</sup> Networking Control Panel	169
10.3.2	Configuring Multiple Interfaces	169
10.4	Using Mach <sup>Ten</sup> as a Domain Name Server	170
10.4.1	Configuring Mach <sup>Ten</sup> as a Domain Name Server	170
10.4.2	The Domain Name Resolver	172
10.5	Using Mach <sup>Ten</sup> as a Web Server	173
10.5.1	Importing Macintosh Image Files	173
10.5.2	Multihoming Your Mach <sup>Ten</sup> Web Server	174
10.6	Using Mach <sup>Ten</sup> as a Router	175
10.6.1	Invoking IP Forwarding	176
10.7	Dual Stack Configuration	177
10.7.1	More RAM	177
10.7.2	Cross Talk	177
10.7.2.1	Routing Between Mach <sup>Ten</sup> and OpenTransport	178
10.8	Using Mach <sup>Ten</sup> as a POP Mail Server	180
10.9	Using Electronic Mail	180
10.10	Using FTP	183
10.11	Using Mach <sup>Ten</sup> as an Anonymous FTP Server	183
10.12	Using Telnet	184
10.13	Using a Distributed File System (NFS)	184
10.14	Using Berkeley r-series Commands	185
10.15	Using Serial Line Communications	186
10.15.1	Cabling and Connectors	186
10.15.2	Matching Serial Ports to UNIX Devices	186

10.15.3	Baud Rates Supported	187
10.15.4	General Purpose Interface (GPI) Serial Cable	188
10.15.5	Connecting a Modem to a Mach <sup>Ten</sup> Serial Port	189
10.15.6	Connecting a Terminal	189
10.15.7	Configuring the System to Support an ASCII Terminal	190
10.15.8	Using PPP	190
10.15.8.1	Setting up Mach <sup>Ten</sup> PPP on Your Macintosh	191
10.15.8.2	Configuring your Mach <sup>Ten</sup> System as a PPP Server	191
10.15.8.3	Configuring your Mach <sup>Ten</sup> System as a PPP Client	194
10.15.8.4	IP Addressing Strategy for Multiple PPP Clients	196
10.15.9	Using SLIP	197
10.15.9.1	Configuring Mach <sup>Ten</sup> to Dial Out to SLIP Servers	197
10.15.9.2	Tip Activity Log	198
10.15.9.3	The Tip Configuration Database	198
10.15.9.4	Hardware Flow Control	200
10.15.9.5	Tip Login Script	201
10.15.9.6	Header Compression	202
10.15.9.7	Setting a Default Route	202
10.15.9.8	Setting the MTU	203
10.15.10	Configuring Mach <sup>Ten</sup> as a Dial-In SLIP Server	203
10.15.10.1	Serial Port Set Up	203
10.15.10.2	Login Authentication	204
10.15.10.3	The Sliplogin Program	205
10.15.10.4	SLIP Session Termination	205
10.15.10.5	Manual Connections to SLIP Servers	206

---

10.15.10.6	Logging In	207
10.15.11	Starting SLIP	207
10.15.11.1	Testing Connectivity	208
10.15.11.2	Stopping SLIP	209
10.15.11.3	C-Shell Scripts to Expedite SLIP Connections	210
<b>11.0</b>	<b>Mach<sup>Ten</sup> Programming Environment</b>	<b>213</b>
11.1	Mach <sup>Ten</sup> Development Tools	213
11.1.1	Programs, Libraries and Include Files	213
11.1.2	Documentation	217
11.1.3	Program Sources	218
11.2	PEF and XCOFF	219
11.3	Shared Libraries	219
11.3.1	Shared Library Production	220
11.3.2	Run-Time and Compile-Time Libraries	220
11.3.3	Run-Time and Compile-Time Naming Conventions	221
11.4	Traditional UNIX Libraries	221
11.5	Header Files	222
11.5.1	Pre-Defined Names	222
11.6	Compiling Sources	224
11.6.1	Ada	224
11.6.2	C	225
11.6.3	Objective-C	226
11.6.4	C++	226
11.6.5	Fortran	227
11.6.6	Java	228
11.7	Linking Executables	229

---

11.7.1	ld	229
11.7.2	mkpef	229
11.8	To make or pmake	230
11.9	Symbol Information	230
11.10	Debugging	231
11.10.1	Debugging using gdb	231
11.10.2	Macintosh Debugging Tools	232
11.10.2.1	MacsBug	232
11.10.2.2	Macintosh Debugger for PowerPC	233
11.10.2.3	Metrowerks Debugger	233
11.10.3	Environment Variables for Debugging and Monitoring	234
11.10.3.1	DEBUGGERFIRST	235
11.10.3.2	STACKCHK	235
11.10.3.3	MEMSTATS	235
11.11	Making Macintosh Applications	236
11.11.1	Macintosh OS Header Definition Files	236
11.11.2	Macintosh OS Interface Libraries	237
11.11.3	Macintosh Application Startup Routine	238
11.11.4	Macintosh Application Construction	239
11.12	Cross-Development Tools and Targets	240
11.12.1	Default Mach <sup>Ten</sup> Environment	240
11.13	Porting Software to Mach <sup>Ten</sup>	241
11.13.1	Real Memory Issues	241
11.13.1.1	Stack Overrun	241
11.13.1.2	Allocating Memory in Mach <sup>Ten</sup>	242
11.13.1.3	Calculating Memory Requirements	243
11.13.1.4	Setting the Mach <sup>Ten</sup> Heap Size	244
11.13.1.5	Problem Areas	245

---

11.14	Programming Example	247
11.14.1	Rogue	247
11.14.2	Building the Executable	247
11.14.3	Debugging Using MacsBug or Other Macintosh Debuggers	248

## **12.0 The X Window System 249**

12.1	The X Desktop	250
12.1.1	Starting the X Server	250
12.1.2	The Menu Bar	251
12.1.2.1	The File Menu	252
12.1.2.2	The Window Menu	253
12.1.3	Automatic Launch of the X Server	255
12.1.4	Quitting the X Server	255
12.1.5	Running the X Server Without a Menu Bar	255
12.1.5.1	Menu Bar Shortcuts	255
12.2	Administering the X Window Software Environment	257
12.2.1	Starting Clients	257
12.2.1.1	The Startup Script	257
12.2.1.2	Resources — X Application Preferences	260
12.2.2	The Window Manager Client	266
12.2.2.1	Starting the Window Manager	266
12.2.2.2	Using the Mouse	267
12.2.2.3	Selecting a Window	267
12.2.2.4	Moving a Window	268
12.2.2.5	Changing the Size of a Window	270
12.2.2.6	Changing a Window Into an Icon	272
12.2.2.7	Moving an Icon	273

---

	12.2.2.8	Restoring a Window from an Icon	273
	12.2.2.9	Displaying a Window Menu and Making Selections	274
	12.2.2.10	Summary of Window Menu Functions	275
	12.2.2.11	Raising a Window	276
	12.2.2.12	Quitting the Window Environment	277
12.2.3		The X Server Program	278
	12.2.3.1	X Server Startup Options	278
	12.2.3.2	Mouse Button Mapping	281
	12.2.3.3	Keyboard Mapping	284
	12.2.3.4	Server Error Logging	285
	12.2.3.5	X Server Performance Tuning Guide	286
	12.2.3.6	The Default Font Path	287
	12.2.3.7	Obtaining Fonts from a Network Font Server	288
	12.2.3.8	Providing Fonts Over the Network	289
12.2.4		X Display Management Under Mach <sup>Ten</sup>	289
	12.2.4.1	The Mach <sup>Ten</sup> X Server and XDM	290
12.2.5		X Over Serial Lines	293
12.3		Mach <sup>Ten</sup> X Window Software Overview	294
	12.3.1	Preparing Your Macintosh Control Panels	296
	12.3.2	Getting Started With X	299
	12.3.3	Building X Applications	299
	12.3.3.1	Running X Client Applications	301
	12.3.3.2	The X11 Application Development Environment Under Mach <sup>Ten</sup>	301
	12.3.3.3	Programming Notes	303



---

## LIST OF FIGURES

---

Figure 1.	Mach <sup>Ten</sup> on the Macintosh Desktop	3
Figure 2.	Mach <sup>Ten</sup> Terminal Window Desktop	4
Figure 3.	xterm Displayed by the AfterStep Window Manager	5
Figure 4.	Mach <sup>Ten</sup> System Architecture	8
Figure 5.	Mach <sup>Ten</sup> CD-ROM Contents	16
Figure 6.	Mach <sup>Ten</sup> Folder	17
Figure 7.	Mach <sup>Ten</sup> Console Window	19
Figure 8.	Chnglicense Dialog Box	20
Figure 9.	Control Panel Warning Dialog Box	32
Figure 10.	Mach <sup>Ten</sup> Control Panel	33
Figure 11.	General Configuration Screen	35
Figure 12.	Memory Configuration Screen	36
Figure 13.	Networking Configuration Screen	38
Figure 14.	Example Network Configuration	43
Figure 15.	Mach <sup>Ten</sup> Application Folder	45
Figure 16.	Mach <sup>Ten</sup> Initialization	46
Figure 17.	Mach <sup>Ten</sup> Login Console	50
Figure 18.	Mach <sup>Ten</sup> Terminal Window	51
Figure 19.	Mach <sup>Ten</sup> Windows	52
Figure 20.	A .cshrc File	76
Figure 21.	The /etc/passwd File	78
Figure 22.	The /etc/group File	79
Figure 23.	Special Keyboard Characters	83
Figure 24.	Using ps to Show What is Running	84
Figure 25.	Example of File Permissions	101
Figure 26.	Spooling Printer Output to UNIX Printers Using lwsrv	158

---

Figure 27.	Resource Sharing: FTP, mail, rlogin	163
Figure 28.	Example /etc/hosts File	171
Figure 29.	General Purpose Interface (GPI) Serial Cable	188
Figure 30.	The X Desktop	250
Figure 31.	The File Menu	252
Figure 32.	The Window Menu	253
Figure 33.	The Order Sub-Menu	254
Figure 34.	The OpenLook Environment	266
Figure 35.	Grabbing the Title Bar	268
Figure 36.	Repositioning the Window	269
Figure 37.	A Window and Its Elastic Outline	271
Figure 38.	The Title Bar Minimize Button	272
Figure 39.	The OpenLook Icon	272
Figure 40.	The Restored Window	273
Figure 41.	The OpenLook Window Menu	274
Figure 42.	A Partially Concealed Window	276
Figure 43.	The Window Revealed	277
Figure 44.	Logitech MouseMan <sup>TM</sup> Control Panel	283
Figure 45.	Monitors & Sound Control Panel	296
Figure 46.	Mouse Control Panel	297
Figure 47.	Keyboard Control Panel	297
Figure 48.	Control Strip Control Panel	298
Figure 49.	Mach <sup>Ten</sup> Control Panel	298

## 1.0 Power Mach<sup>Ten</sup> — UNIX on the Macintosh Desktop

Welcome to Mach<sup>Ten</sup>, a complete UNIX<sup>†</sup> OS that runs as a Macintosh application. Mach<sup>Ten</sup> 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 to Macintosh applications as well. Mach<sup>Ten</sup> also supports a sophisticated set of UNIX and X software development tools, a high-performance X Window server, and hundreds of UNIX applications.

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.

Mach<sup>Ten</sup> includes a complete suite of C, C++, Objective-C, Fortran, Ada and Java development tools. Mach<sup>Ten</sup> can be used in combination with standard Macintosh editors and compilers to develop UNIX applications, X applications and Macintosh applications, or to develop hybrid Macintosh/UNIX applications. Mach<sup>Ten</sup>'s PowerPC compiler suite can be used to recompile existing UNIX applications or to develop new UNIX applications. And the resultant applications can be turned into clickable Macintosh applications.

This unique toolset from Tenon gives developers the ability to create an application with a single source base not only for Power Macintoshes under a native Apple operating system, but also for Silicon Graphics machines, SUNs, DEC, or HP workstations. Mach<sup>Ten</sup> gives developers the freedom to take advantage of time-tested UNIX development tools and to experiment with Java or Ada tasking, without giving up the features of their favorite Macintosh editors. Mach<sup>Ten</sup> creates a new standard in PowerPC software development.

---

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

Mach<sup>Ten</sup>'s development environment includes the GNU compiler for Ada95, the first internationally standardized object-oriented language. Mach<sup>Ten</sup>'s Macintosh Toolbox bindings let you use Ada95 to produce native Macintosh applications. In addition, Mach<sup>Ten</sup> includes the latest GNU compilers for C, C++, Objective-C and Fortran, and the latest GNU internet programming tools, such as Perl, tcl/tk and expect. The Kaffe Java virtual machine lets you run your newly developed Java bytecode. Because all Mach<sup>Ten</sup> compilers generate standard PowerPC binary formats, Mach<sup>Ten</sup> can be used in combination with any standard Macintosh compiler and debugger. And, just in case you prefer Macintosh tools, we've included such favorites as MacPerl, Alpha and BBEditLite.

## 1.1 The Mach<sup>Ten</sup> Desktop

The Macintosh “desktop” is created and maintained by a Macintosh system application called Finder<sup>™</sup>. When Mach<sup>Ten</sup> is installed on your Macintosh, you still have the Finder desktop available, and you can easily switch from Macintosh applications to Mach<sup>Ten</sup>. Mach<sup>Ten</sup> and MacOS essentially operate as co-resident operating systems, sharing the processor and being enriched by each other's environment. Using Mach<sup>Ten</sup> 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. Mach<sup>Ten</sup> on the Macintosh Desktop” shows a typical Mach<sup>Ten</sup> desktop display.

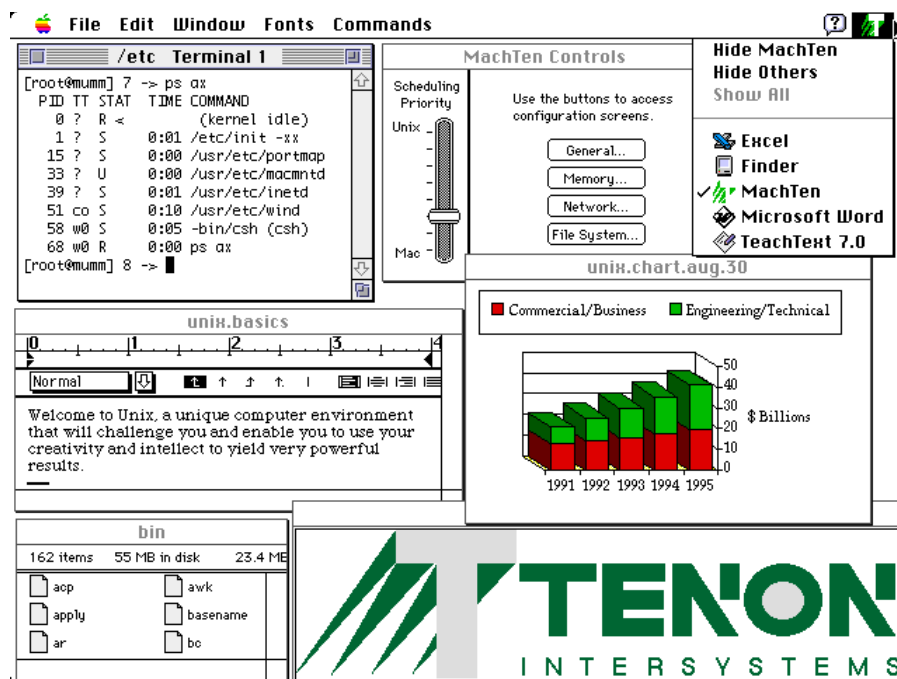


Figure 1. Mach<sup>Ten</sup> on the Macintosh Desktop

## 1.1.1 Mach<sup>Ten</sup> Terminal Window Desktop

Mach<sup>Ten</sup>'s terminal window desktop displays a customizable set of independent command windows. “Figure 2. Mach<sup>Ten</sup> Terminal Window Desktop” shows a Mach<sup>Ten</sup> desktop display with multiple terminal windows on the Finder desktop. If you log in as “root” or “mtuser” (with “MachTen” as the password), the UNIX terminal window will be displayed.

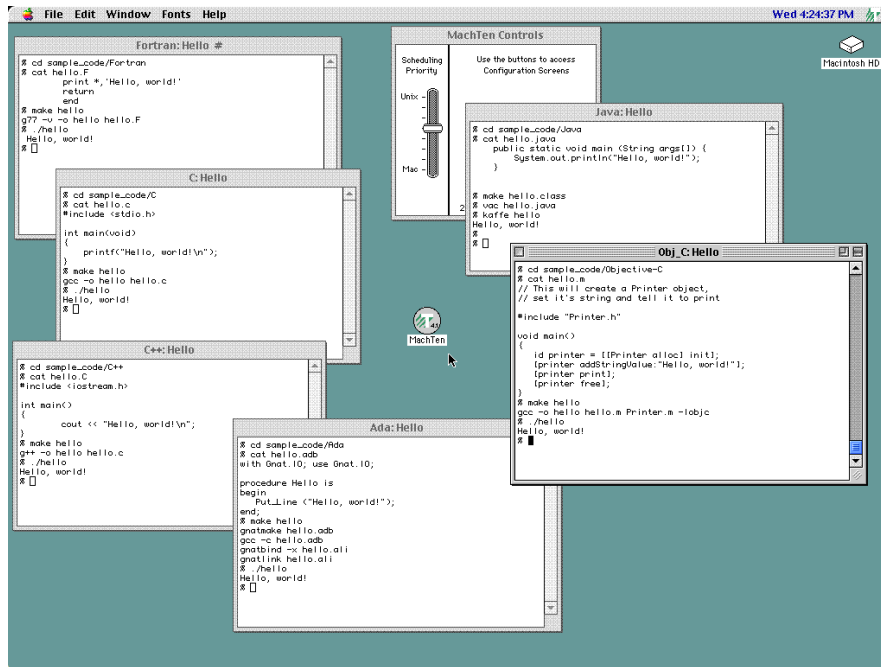
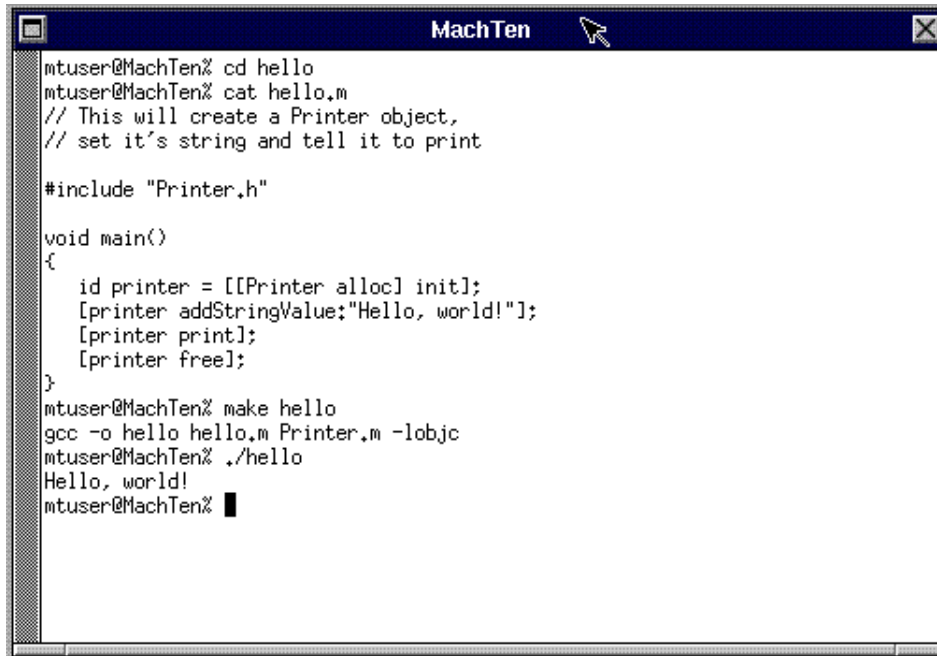


Figure 2. Mach<sup>Ten</sup> Terminal Window Desktop

Mach<sup>Ten</sup> also supports an X desktop. A variety of X window managers are available. If you log in to Mach<sup>Ten</sup> as "mtnext" (with "MachTen" as the password), you will automatically run the AfterStep X window manager, which displays a NeXT-like desktop.



```
mtuser@MachTen% cd hello
mtuser@MachTen% cat hello.m
// This will create a Printer object,
// set it's string and tell it to print

#include "Printer.h"

void main()
{
    id printer = [[Printer alloc] init];
    [printer addStringValue:"Hello, world!"];
    [printer print];
    [printer free];
}
mtuser@MachTen% make hello
gcc -o hello hello.m Printer.m -lobjc
mtuser@MachTen% ./hello
Hello, world!
mtuser@MachTen% █
```

Figure 3. xterm Displayed by the AfterStep Window Manager

## 1.2 The Mach<sup>Ten</sup> “UNIX Virtual Machine”

Mach<sup>Ten</sup> 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<sup>Ten</sup> 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<sup>Ten</sup>, Macintosh applications, UNIX applications, Mach applications and X applications run simultaneously. Mach<sup>Ten</sup> represents the confluence of the best features of the Macintosh and the systematic power and elegance of UNIX and Mach.

Tenon's Mach<sup>Ten</sup> technology is an extension to MacOS. Using standard Macintosh shared libraries and code fragments, Tenon's Mach<sup>Ten</sup> extends MacOS to create a UNIX virtual machine. The UNIX virtual machine (UVM) is implemented within the context of a standard Macintosh application program. Therefore, other existing Macintosh applications run in parallel with the UVM in a highly compatible fashion. The UVM, in turn, implements a standard UNIX API (POSIX) for a large family of traditional UNIX commands and utilities. The UVM also implements a fully pre-emptive execution environment for its applications. Since each UNIX application program is implemented as a Macintosh code fragment, UNIX applications are essentially “UNIX plug-ins”. So, Mach<sup>Ten</sup> is a highly-portable UNIX virtual machine that implements its applications as UNIX plug-ins running within a Macintosh application context.



## 1.3 The Mach<sup>Ten</sup> Architecture

“Figure 4. Mach<sup>Ten</sup> System Architecture” shows the basic system architecture. Mach<sup>Ten</sup> consists of a family of shared libraries — Mach<sup>Ten</sup> itself is a collection of shared libraries; the traditional UNIX libraries (*libc*, *libm*, etc.) are each a shared library, and every UNIX application is also shared. This organization takes advantage of the shared library, dynamic linking and memory mapped file access features of the MacOS, and maximizes memory savings, as each software component is loaded into memory only once. Tenon's native fast file system (FFS) gives the Mach<sup>Ten</sup> development tools access to UNIX-level performance and features.

### 1.3.1 Dynamic Memory Configuration

Mach<sup>Ten</sup> 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.

### 1.3.2 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 run-time version of the library is used to satisfy library calls. If that library has already been loaded into memory, access is obtained to the run-time library through a dynamic linkage process. Otherwise, the library is loaded into memory and 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.

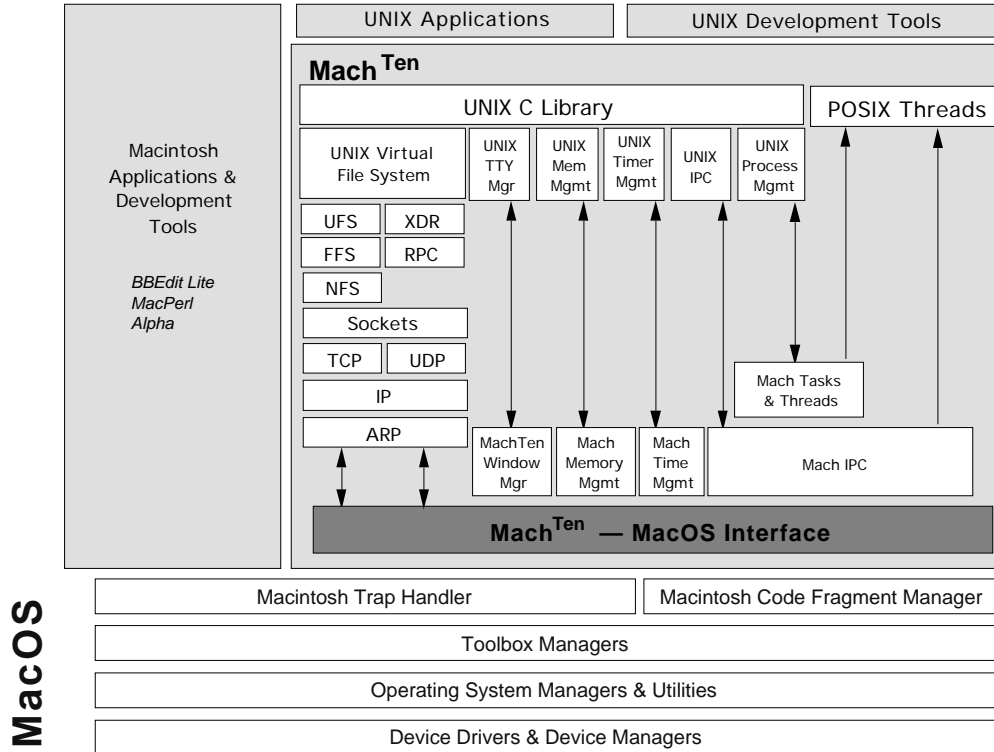


Figure 4. Mach<sup>Ten</sup> System Architecture

### 1.3.3 Memory Mapped File Access

The Power Macintosh Memory Manager provides memory mapped file 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.

### 1.3.4 Integrated Software Development Tools

Power Mach<sup>Ten</sup> 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<sup>Ten</sup> development system. With Mach<sup>Ten</sup>, 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<sup>Ten</sup> system requests. Such hybrid applications will be able to simultaneously take advantage of the power of the MacOS and UNIX.

### 1.3.5 Native Fast File System

Mach<sup>Ten</sup> uses a derivative of Berkeley's BSD Fast File System to dramatically increase speed and save space. The Fast File System circumvents the limitations imposed by the Macintosh Hierarchical File System. More information about the Fast File System (FFS) is available in section “6.0 The Mach<sup>Ten</sup> File Systems”.



## 2.0 Installing Mach<sup>Ten</sup>

Mach<sup>Ten</sup> will run on all Power Macintoshes with MacOS version 7.1.2 or later. The entire Mach<sup>Ten</sup> system is contained on a CD-ROM and is comprised of the following components:

### Mach<sup>Ten</sup> README and Online User's Guide

This component contains two README files — *README.html* and *README*. The *README* file is strictly a text file and can be displayed by simply double-clicking on the file and scrolling through the document. The *README.html* is meant to be displayed by a browser such as Netscape Communicator or Internet Explorer. The *README* files contain up-to-the-minute release information and directions that became available after the documentation was frozen for production purposes.

### Licensing

This folder contains important licensing information concerning Mach<sup>Ten</sup> and various software subsystems that have been licensed from other parties and included with this release. It is important that you review and agree with the licensing information contained in this directory and that you abide by the covenants and restrictions found therein.

### Mach<sup>Ten</sup> Installer

This is an installer that contains the Mach<sup>Ten</sup> application, the Mach<sup>Ten</sup> virtual machine and the UNIX libraries. The standard Mach<sup>Ten</sup> application set includes network access programs, printer spooling, text editors, basic utilities, administration aids and software development tools. The biggest part of the installation involves double-clicking on this installer and allowing it to transfer a Mach<sup>Ten</sup> image to a local hard disk.

## Documentation

This directory contains the data files that compose the online electronic documentation. This documentation includes an electronic Mach<sup>Ten</sup> User's Guide, UNIX User, Programmer and System Administrator documents, GNAT documentation in PostScript for the Mach<sup>Ten</sup> Ada environment, and an Objective-C overview.

## Utilities

Contains an *index.html* file which is meant to be viewed with an internet browser. The index file contains information about four sub-folders — Debuggers, Development Utilities, Text Utilities and Fast File Utilities. The index file describes the contents of each of the sub-folders and their use within the Mach<sup>Ten</sup> environment.

### **index.html**

This HTML file provides an overview of the contents of the *Utilities* folder.

### **Debuggers**

Contains Apple's MacsBug, Macintosh Debugger, and information on obtaining The Debugger (also known as Jasik).

### **Development Utilities**

Contains the Ada 95 Mac Application Framework (AMAF), libraries and *include* files from Essential Tools and Objects (ETO) #22, and Apple's resource editor ResEdit.

### **Text Utilities**

Contains the Alpha<sup>†</sup>, Adobe Acrobat and BBEditionLite text editing and presentation applications. It also contains *Unix <-> Text*, an application for converting the end-of-line characters in a text file between standard Macintosh and UNIX conventions.

---

<sup>†</sup> Alpha is shareware and is not registered. If you use Alpha, please register it.

## Fast File Utilities

Contains tools to aid in fast file system maintenance. *Fast File First Aid* is the Mach<sup>Ten</sup> *fsck* utility built as a Macintosh application. It performs an interactive file system check and repair operation. *Fast File First Aid* may be used to correct file system corruption which can prevent Mach<sup>Ten</sup> from booting up. The *Recovery* folder contains a special minimal version of Mach<sup>Ten</sup> for use when file system corruption has affected key programs or support files. In the unlikely event that file corruption occurs, drag the *Recovery* folder onto a hard drive and launch the *MachTen.emerg* application. This will allow you to access your file systems and take appropriate actions from within a working Mach<sup>Ten</sup> environment.

## FFS\_Installer

Mach<sup>Ten</sup> includes an installer utility called *FFS\_Installer* to greatly simplify the task of adding fast file systems to your Mach<sup>Ten</sup> system.

To create a new FFS, launch this program and select the desired size FFS. Install it into the *MachTen* folder and launch Mach<sup>Ten</sup>. The new FFS will be automatically mounted when Mach<sup>Ten</sup> starts up.

## Mac\_Perl\_510r2\_appl

This application installs MacPerl. Mach<sup>Ten</sup> also includes UNIX Perl.

## OpenGL Installer

This application installs OpenGL capabilities into Mach<sup>Ten</sup>. To receive the necessary OpenGL license number, you must separately purchase the OpenGL option.

## Source\_FFS

This is a large file of about 400MB of data. It is a Mach<sup>Ten</sup> fast file system file that is meant to be mounted within the Mach<sup>Ten</sup> file system under the Mach<sup>Ten</sup> directory. If the Mach<sup>Ten</sup> CD-ROM is inserted when Mach<sup>Ten</sup> is started, the *Source\_FFS* fast file system will be automatically mounted and immediately available by reference to files and directories under the */base/src* directory. The *Source\_FFS* contains a great deal of the source that goes into making Mach<sup>Ten</sup> and its applications.

Mach<sup>Ten</sup> runs as an application, therefore all of the network and peripheral devices that your Macintosh supports are still supported when Mach<sup>Ten</sup> is installed. Since Mach<sup>Ten</sup> internet software has been interfaced to MacOS LocalTalk, MacOS EtherTalk, MacOS TokenTalk and MacOS serial port hardware, all of the Mach<sup>Ten</sup> 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<sup>Ten</sup> software package.

## 1. Site Preparation

- **Verify RAM**

Pull down the Apple menu and choose "About this Computer". 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 32MB<sup>†</sup>. Check MacOS version and "Power Macintosh".

- **Verify disk space**

Make sure the hard disk has at least 350MB available by viewing any folder on your hard disk using "View by Icon". If there is not ample free disk, you will need to remove documents and folders to make additional space on your hard disk. Mach<sup>Ten</sup> does not have to be installed on the MacOS boot volume. Much of Mach<sup>Ten</sup> is contained within Mach<sup>Ten</sup> fast file system files. These are UNIX file systems that exist within MacOS files. See section "6.3 Mach<sup>Ten</sup> FFS" for a complete discussion of Mach<sup>Ten</sup> fast file systems.



**The free disk space requirement numbers are based on a target volume with an unformatted capacity of 1GB. 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 1GB capacity disk will require up to 450MB for Mach<sup>Ten</sup>. To conserve disk space, we suggest installing Mach<sup>Ten</sup> on HFS+ partitions.**

---

<sup>†</sup> Mach<sup>Ten</sup> will run with 32MB of RAM. However, 48MB of RAM is recommended for good performance. Power Mach<sup>Ten</sup> itself consumes about 14MB of RAM during normal operation (excluding X Windows).



- **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. Mach<sup>Ten</sup> works with HFS+ partitions.

All MacOS system software, network drivers and other peripheral software should be installed before the Mach<sup>Ten</sup> application is started.

The Mach<sup>Ten</sup> Installer adds extensions and control panels to your System Folder, so make sure your System Folder is not protected during the installation.

If you are installing Power Mach<sup>Ten</sup> over an existing version of Mach<sup>Ten</sup> on your disk, you may want to review section "2.2 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<sup>Ten</sup> 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<sup>Ten</sup> includes its own TCP protocol stack. If you have already configured the Apple TCP/IP control panel, Mach<sup>Ten</sup> will use the networking information provided by OpenTransport by default.

If you are running Mach<sup>Ten</sup> on a Macintosh that is not connected to a network, you can reconfigure Mach<sup>Ten</sup> for the added network connections at a later time. Refer to section “2.2.1 Reconfiguration”.

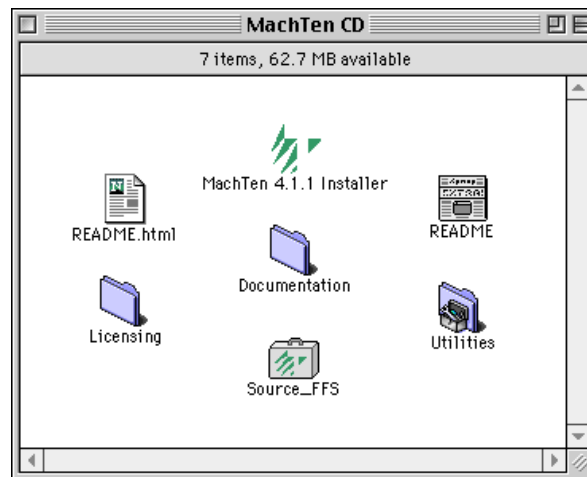
- **Quit all applications**

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.

## 2. Install the Mach<sup>Ten</sup> Software on your Hard Disk

- **Insert the Mach<sup>Ten</sup> CD-ROM**

Insert the Mach<sup>Ten</sup> CD-ROM and double-click on the disk icon. The disk will show an open window as illustrated below.

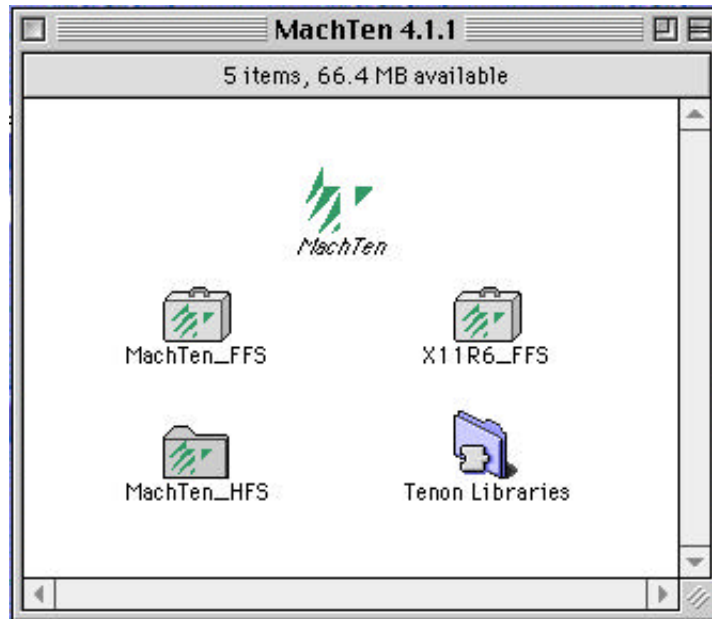


**Figure 5. Mach<sup>Ten</sup> CD-ROM Contents**

The installation procedure requires double-clicking on the Mach<sup>Ten</sup> Installer and following the directions contained there. The following steps detail the installation instructions.

- **Double-click on the Mach<sup>Ten</sup> Installer**

Double-click on the *MachTen 4.x Installer* on the CD-ROM. This will start a self-extracting operation that will create a *MachTen 4.X* folder on the hard drive that you select. Additionally, it will install a number of files in the active System Folder supporting control panel and System Folder extensions. This will require approximately 350MB of disk space and may take between 10 and 15 minutes.



**Figure 6. Mach<sup>Ten</sup> Folder**



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<sup>Ten</sup> is dependent on its file system hierarchy, including the names, locations and permissions of its files.

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

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

- **Reboot**

After the installation process is complete, the installer application will request that it be allowed to reboot the system. When given an OK, the system will be automatically rebooted. This process is necessary to incorporate the Mach<sup>Ten</sup> control panel and Mach<sup>Ten</sup> system extensions.

- **Configure the Mach<sup>Ten</sup> control panel**

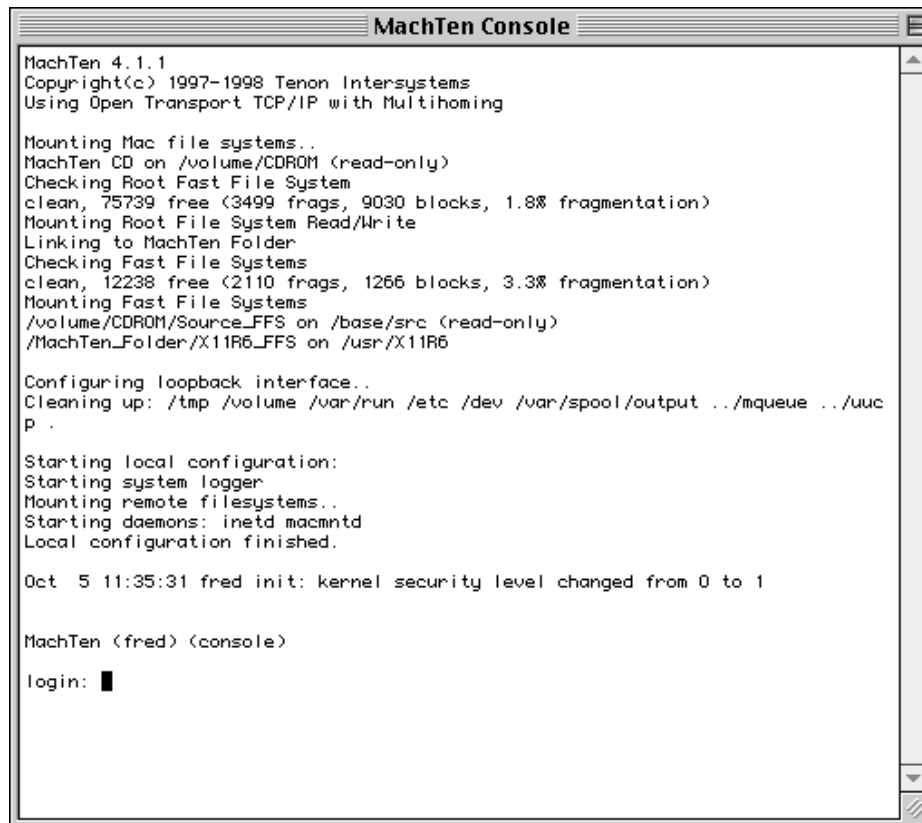
Configure the Mach<sup>Ten</sup> control panel according to the instructions in section "3.0 The Mach<sup>Ten</sup> Control Panel".

### 3. Launch Mach<sup>Ten</sup>

- **Double-click on the Mach<sup>Ten</sup> application icon**

Launch Mach<sup>Ten</sup> by double-clicking on the *MachTen* application icon in the *MachTen* folder. For easier launching, you may make an alias to this application and put it on the desktop or in the *System Folder/Apple Menu Items* folder. To automatically start Mach<sup>Ten</sup> when your Macintosh is started, you may put an alias in the *System Folder/Startup Items* folder. If your system does not successfully start up and present you with the "login:" prompt after a few moments, contact Tenon Technical Support at 1-800-6-Mach10.

A Mach<sup>Ten</sup> console window will appear with system configuration information taken from the Mach<sup>Ten</sup> control panel.



```
MachTen Console
MachTen 4.1.1
Copyright(c) 1997-1998 Tenon Intersystems
Using Open Transport TCP/IP with Multihoming

Mounting Mac file systems..
MachTen CD on /volume/CDROM (read-only)
Checking Root Fast File System
clean, 75739 free (3499 frags, 9030 blocks, 1.8% fragmentation)
Mounting Root File System Read/Write
Linking to MachTen Folder
Checking Fast File Systems
clean, 12238 free (2110 frags, 1266 blocks, 3.3% fragmentation)
Mounting Fast File Systems
/volume/CDROM/Source_FFS on /base/src (read-only)
/MachTen_Folder/X11R6_FFS on /usr/X11R6

Configuring loopback interface..
Cleaning up: /tmp /volume /var/run /etc /dev /var/spool/output ../mqueue ../uuc
p .

Starting local configuration:
Starting system logger
Mounting remote filesystems..
Starting daemons: inetd macmtd
Local configuration finished.

Oct  5 11:35:31 fred init: kernel security level changed from 0 to 1

MachTen (fred) (console)

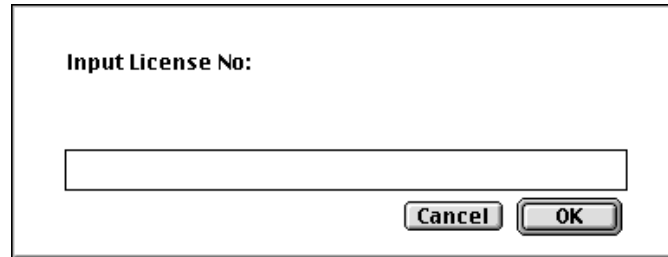
login: █
```

**Figure 7. Mach<sup>Ten</sup> Console Window**

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

#### 4. Enter Your License Number

Mach<sup>Ten</sup> license numbers can be entered at any time before the temporary license expires. To enter your license number, execute the *chnglicense* program and type your license number into the dialog box. Be careful to use proper case for the letters and include the dashes ('-') in the license number.



**Figure 8. Chnglicense Dialog Box**

Click "OK" to complete the license number registration process.

## 2.1 Performance Tuning — Optimizing Mach<sup>Ten</sup>

Mach<sup>Ten</sup> differs from other UNIX implementations in several significant ways. Mach<sup>Ten</sup> 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<sup>Ten</sup> performance by tailoring your system to fit your system configuration. Mach<sup>Ten</sup> 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<sup>†</sup>) and adhering to the guidelines of disk optimizing tools will improve the Mach<sup>Ten</sup> system's performance.

- **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.

- **Turn off file sharing**

Disable the file sharing option located in the File Sharing control panel.

- **Close control panels after use**

Do not keep the Mach<sup>Ten</sup> Scheduling Priority slide bar or any control panel on the desktop. An open control panel consumes valuable CPU cycles.<sup>††</sup>

---

<sup>†</sup> See <http://www.fwb.com> or phone (650)482-4800.

<sup>††</sup> Refer to section "3.0 The Mach<sup>Ten</sup> Control Panel" for a more details on using the Scheduling Priority slide bar.

- **Adjust your search path**

When running Mach<sup>Ten</sup> in the default configuration, the following directories are searched (in the order listed) when a Mach<sup>Ten</sup> command is executed:

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

If the user is *root*, these two additional directories are searched:

```
/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<sup>Ten</sup> 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.

## 2.1.1 System Optimization Guidelines

The System Optimization Guidelines table which follows will help you customize Mach<sup>Ten</sup> systems for particular environments. A little system attention will go a long way to give you a smoothly running system.

It is important to remember that Mach<sup>Ten</sup> 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<sup>Ten</sup>, even with limited memory.



Table 1. System Optimization Guidelines

<p><b>Disk Space Limitations</b></p> <p>A system with 500MB or less of hard disk will probably require careful attention to “tidiness” of the available disk space.</p>	<p>By default, the <i>/etc/rc</i> file will preserve the editor files, and then automatically remove files in <i>/tmp</i> on each restart. (Note that subdirectories in <i>/tmp</i> are not purged.)</p> <p>A <i>cron(8)</i> entry may be created to purge old ‘preserved’ editor files from <i>/var/preserve</i>.</p> <p>If a server system is available with more disk space, always use NFS.</p> <p>Use <i>df(1)</i> to determine the capacity and available space of each mounted file system.</p>
<p><b>Memory Space Limitations</b></p> <p>Since Mach<sup>Ten</sup> 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 (<i>execve(2)</i>) will fail with an ENOMEM (Error: not enough memory).</p>	<p>Run only the daemon processes that are needed. If you do not need the services provided by a daemon, you can disable it via <i>rc.conf(5)</i>.</p> <p>Use <i>ps(1)</i> to find out how much memory each process is consuming.</p> <p>If available memory is filled, terminate unwanted processes with <i>kill(1)</i>. This will free up the memory those processes are consuming.</p> <p>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.)</p> <p>You can run many Macintosh applications at the same time by reducing each Macintosh application’s memory allotment. Under Finder, 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.</p>

<p><b>Daemon Processes</b></p> <p>NFS Server: This service is provided by three daemons — <i>portmap</i>, <i>mountd</i> and <i>nfsd</i>.</p>	<p>Run NFS daemons on NFS servers only. There is no advantage to running multiple <i>nfsd</i>'s under Mach<sup>Ten</sup>; they only consume more memory.</p>
<p><b>sendmail</b></p>	<p><i>sendmail</i> 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 <i>rlogin</i> to another machine or use a POP client to read your mail.</p>
<p><b>syslogd</b></p>	<p>This is the system logger daemon. <i>Syslogd</i> 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 <i>sendmail</i> activities. Run <i>syslogd</i> while changing system configuration files for other daemons, or when running <i>sendmail</i>, but not otherwise.</p>
<p><b>inetd</b></p>	<p><i>inetd</i> is only needed for incoming or loopback connections. If a machine is not providing <i>telnetd</i>, <i>rlogind</i>, <i>ftpd</i>, or other similar services, <i>inetd</i> does not need to be run. Note that <i>telnet</i>, <i>rlogin</i> and <i>ftp</i> clients (for outgoing connections) can still be run on demand.</p>
<p><b>Router Service</b></p>	<p>No daemons are necessary to provide IP forwarding, although you may choose to run <i>routed</i>. A system that is forwarding IP packets may allocate additional <i>mbufs</i> dynamically, but these allocations are small compared to the size of most programs.</p>
<p><b>Serial Ports</b></p>	<p><i>getty</i> is a process like any other, so an unused serial line with a waiting <i>getty</i> is consuming memory. Ttys can be turned on and off when needed (see <i>init(8)</i>).</p>

<p><b>wind</b></p> <p>The default <i>.profile</i> and <i>.login</i> files automatically start up the window daemon each time a login occurs in the console. Note that the window daemon is started with the command:</p> <pre>exec /usr/libexec/wind</pre> <p>to free up the space of the invoking call.</p>	<p><i>wind(8)</i> is the window daemon controlling the Mach<sup>Ten</sup> window environment. It is not necessary to use the window daemon since Mach<sup>Ten</sup> also supports a simple console device. However, the added value of the window environment is almost always worth the extra memory it consumes.</p>
<p><b>lpd</b></p>	<p>Printing services are provided by <i>lpd(8)</i>. Systems that do not require printing services need not run this daemon. Mach<sup>Ten</sup> will automatically start this daemon when a print command is invoked.</p>
<p><b>httpd</b></p>	<p>Run only if providing a World Wide Web service.</p>

### 2.1.2 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 that 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.

## 2.2 Reinstallation

Occasionally you may want to reinstall Mach<sup>Ten</sup>. Your current system may become damaged, or you may need to install a system upgrade. Perform the following steps before reinstalling Mach<sup>Ten</sup>.

### 1. Preserve Personalized Files and Files That Have Been Changed

- Create a folder named *Preserve* in the top level of your hard disk
- Preserve changed configuration files

Copy 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.

- Copy changed files to the *Preserve* folder

Copy 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

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 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.

### 2. Quit Mach<sup>Ten</sup>

### 3. Remove Outdated Mach<sup>Ten</sup> System Files From Your Hard Disk

- Drag files to the Trash

From the top-level folder, drag the *MachTen* folder to the Trash. Drag the *Tenon Libraries* to the Trash. Certain system configuration information is stored in the Mach<sup>Ten</sup> Preferences. This configuration information will be re-assumed if you do not drag the Preferences to the Trash.

- Empty the Trash

### 4. Install the New Mach<sup>Ten</sup> System

### 5. Restore the Saved Control Panel Settings

- Open the Mach<sup>Ten</sup> control panel

Before launching Mach<sup>Ten</sup>, open the Mach<sup>Ten</sup> control panel and visit each configuration screen. Verify that your installation parameters are correct. Press the “Save” button, even if no changes were made.

### 6. Launch Mach<sup>Ten</sup> by Double-Clicking the Mach<sup>Ten</sup> Application

### 7. Restore the Remaining Saved Configuration Files

- Check files in the *Preserve* and */etc* folders

When the Mach<sup>Ten</sup> re-installation is complete, compare any customized configuration files from the *Preserve* folder to the corresponding files in the */etc* and */home* folders.



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

## 2.2.1 Reconfiguration

If you change your control panel settings, or if you installed Mach<sup>Ten</sup> before you were attached to a network, numerous files must be updated. This may be done automatically by using the Mach<sup>Ten</sup> 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.

### 2.2.1.1 Automated Reconfiguration

In automated configuration, the following files are updated according to the information entered in the Networking Configuration screen of the Mach<sup>Ten</sup> control panel:

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

Take the following actions:

- **Quit Mach<sup>Ten</sup>**
- **Open the MachTen control panel**

Choose “Control Panels” from the Apple menu and open “MachTen Controls”. Press the “Networking” button, make the appropriate changes, and press the “Save” button.

- **Change settings**

Make the appropriate changes in the General or Memory configuration screen, and click the “Save button. Click on “Cancel” to abort the changes (see section “3.0 The Mach<sup>Ten</sup> Control Panel”).

- **Launch Mach<sup>Ten</sup>**

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.
```

### 2.2.1.2 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 (local host change) /etc/sendmail.cw (local host change)
IP address change	/etc/hosts
network service change (for example, disabling <i>ftpd</i> )	/etc/inetd.conf
NFS server change	/etc/fstab /etc/hosts

route change	/etc/hosts /etc/rc.conf
local service change (for example, starting <i>httpd</i> at Mach <sup>Ten</sup> boot up)	/etc/rc.local

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

## 2.2.2 Accessing Mach<sup>Ten</sup> Sources from the CD-ROM

The Mach<sup>Ten</sup> 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<sup>Ten</sup> distribution.

The *Source\_FFS* supplied with Mach<sup>Ten</sup> is 400MB in size. It will be mounted automatically when Mach<sup>Ten</sup> starts up. In order to access files directly from the CD, the CD must be online (visible on the desktop) and be mounted under Mach<sup>Ten</sup>. The default */etc/fstab* file will mount the Mach<sup>Ten</sup> CD on the path */volume/CDROM* if the CD is online when Mach<sup>Ten</sup> is started, or if the CD is inserted when Mach<sup>Ten</sup> is running.

To access the source files from the CD-ROM, a *mount* command must be executed to make the CD-based *Source\_FFS* accessible within the Mach<sup>Ten</sup> file system. As root, execute the following commands:

```
# cd /volume/CDROM
# /sbin/mount -t ffs Source_FFS /base/src
```

Note that the target directory */base/src* 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 first copying them from the CD to your hard disk. Also note that this mount will take place automatically if the Mach<sup>Ten</sup> CD-ROM is installed when Mach<sup>Ten</sup> is started. You can see if the directory */base/src* is already mounted on *Source\_FFS* by issuing a *mount* command with no parameters. This will give you a list of the mounted volumes and the directories on which those volumes are mounted.



## 2.3 Troubleshooting

The Mach<sup>Ten</sup> installation automatically configures the UNIX networking software and 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<sup>Ten</sup>.

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.	<ul style="list-style-type: none"> <li>Temporarily remove or disable extensions, particularly any anti-virus extensions. Unprotect your System Folder. Retry the installation.</li> <li>Check disk space requirements for your installation options.</li> </ul>
Mach <sup>Ten</sup> seems to slow down.	Type the command <code>ps -aux</code> to observe what processes are running. Use <code>kill(1)</code> to terminate unwanted processes. Read section "2.1 Performance Tuning — Optimizing Mach <sup>Ten</sup> " 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 <sup>Ten</sup> file caching, Mach <sup>Ten</sup> 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 or irregular shutdown of Mach <sup>Ten</sup> .	Mach <sup>Ten</sup> must be restarted after a power failure. Traditional UNIX file systems are sensitive to power failures. Use <i>Fast File First Aid</i> in the event of a power failure.

Symptom	Solution
<p>Launching Mach<sup>Ten</sup> causes a system “bomb”.</p>	<p>There may be a control panel or extension conflict. Try temporarily disabling all extensions except the Mach<sup>Ten</sup> extensions to determine whether or not you have a conflict.</p> <ol style="list-style-type: none"> <li>1. Make two new folders in your System Folder called <i>Control Panels check</i> and <i>Extensions check</i>.</li> <li>2. “Select All” in the <i>Control Panels</i> folder, and then hold the &lt;Shift&gt; key down and click “MachTen Controls”. Drag the selected files into the <i>Control Panels check</i> folder.</li> <li>3. “Select All” in the <i>Control Panels</i> folder, and then hold the &lt;Shift&gt; key down and click “MachTen NFS” and “Foreign File Access”. Drag the selected files into the <i>Extensions check</i> folder.</li> <li>4. Restart your Macintosh.</li> <li>5. Run Mach<sup>Ten</sup>.</li> </ol> <p>If Mach<sup>Ten</sup> successfully starts up, a control panel or extension in your system is incompatible with Mach<sup>Ten</sup>. You can identify the culprit by restoring and disabling your control panels and extensions by halves until there is one remaining.</p>
<p>Launching Mach<sup>Ten</sup> causes the Control Panel Warning Dialog Box to appear.</p>	<p>Open the MachTen Controls control panel. Configure and save the General, Memory and Networking screens.</p>

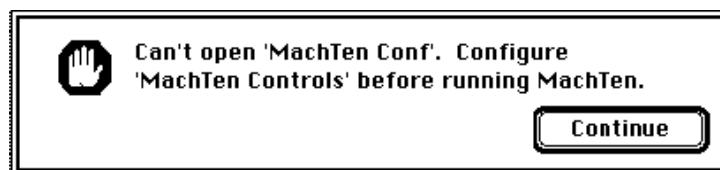


Figure 9. Control Panel Warning Dialog Box

## 3.0 The Mach<sup>Ten</sup> Control Panel

The Mach<sup>Ten</sup> control panel is used to configure certain parameters in your Mach<sup>Ten</sup> system. It is normally opened once prior to launching Mach<sup>Ten</sup> for the first time, and subsequently if you want to change any Mach<sup>Ten</sup> operating parameters.

To view the Mach<sup>Ten</sup> 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<sup>Ten</sup> control panel require you to restart Mach<sup>Ten</sup> in order for the modifications to take effect.

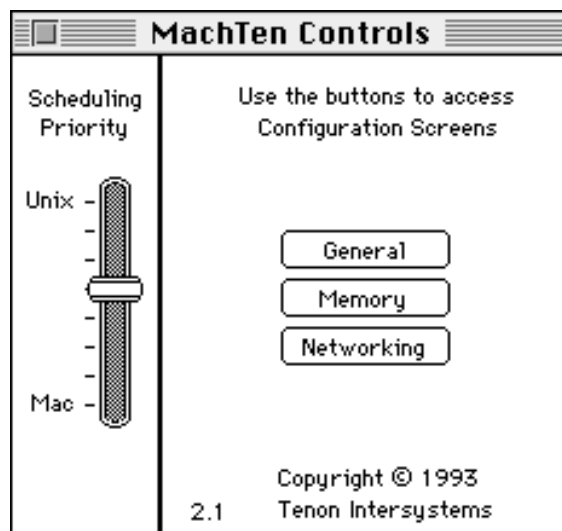


Figure 10. Mach<sup>Ten</sup> Control Panel

## 3.1 Scheduling Priority Slide Bar

On a busy system where both Mach<sup>Ten</sup> 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 optimal performance, the Mach<sup>Ten</sup> control panel should be closed after making any scheduling priority adjustments.

The Mach<sup>Ten</sup> 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<sup>Ten</sup>. Similarly, when no Mach<sup>Ten</sup> processes are active, the impact of Mach<sup>Ten</sup> 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<sup>Ten</sup> to operate properly. These include general Mach<sup>Ten</sup> system information, settings that affect Mach<sup>Ten</sup> 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 main 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 automatically take effect the next time you start Mach<sup>Ten</sup>.

## 3.2.1 General Configuration Screen

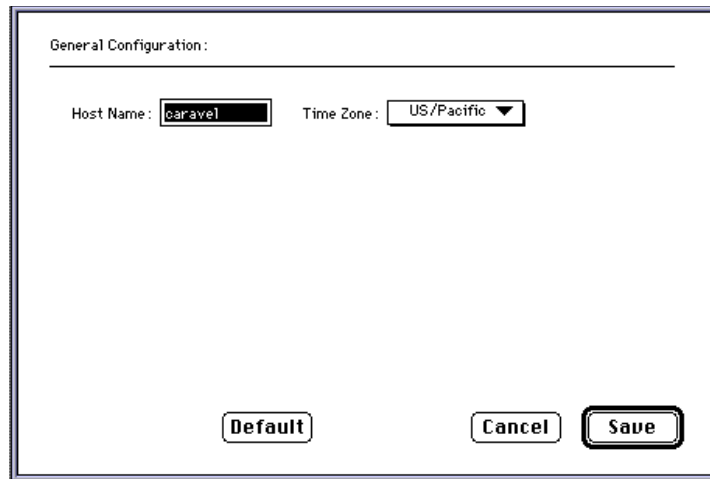


Figure 11. General Configuration Screen

### 3.2.1.1 Host Name

The host name is the name given to your Macintosh. The host name entered will be used in conjunction with addressing information from the Networking Configuration screen (described in section “3.2.3 Networking Configuration Screen”) 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<sup>Ten</sup>, 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<sup>Ten</sup> uses the time as set in the Macintosh Date & Time control panel, typically local time. The time zone information helps Mach<sup>Ten</sup> convert between local time and GMT. Users are responsible for adjusting the Date & Time 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 Map control panel.

### 3.2.2 Memory Configuration Screen

Mach<sup>Ten</sup> uses maximum parameter values in the Memory Configuration screen to dynamically allocate UNIX kernel memory resources at boot time. As your system requirements change, these values can be adjusted accordingly.

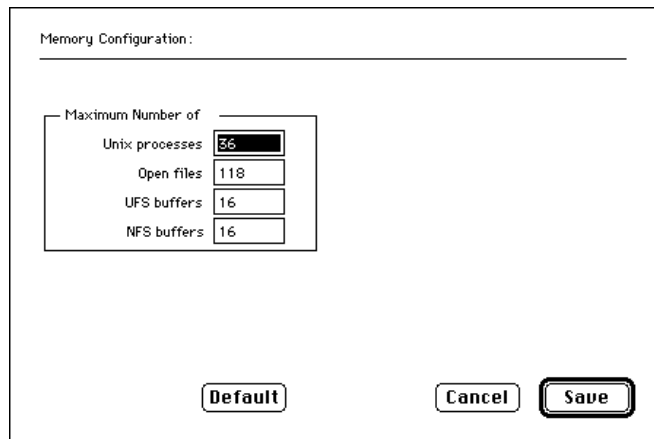


Figure 12. Memory Configuration Screen

### 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<sup>Ten</sup> can have open at one time. 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 increased 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 increased memory usage.

### 3.2.3 Networking Configuration Screen

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

Figure 13. Networking Configuration Screen

Mach<sup>Ten</sup> will extract all default network parameters from the TCP control panel, if available.

See section “10.0 Networking with Mach<sup>Ten</sup>” for more details on internetworking topics introduced by the Networking Configuration screen.

<sup>†</sup> 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 this box to enable forwarding of Internet Protocol (IP) 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

Check this box if you want to receive internet network mail messages directly on your system (i.e., turn your Mach<sup>Ten</sup> system into a mail server). When checked, the UNIX *sendmail* daemon is started automatically when Mach<sup>Ten</sup> is launched. If you use any other mail program (e.g., QuickMail, ccMail, Eudora), leave this box unchecked. By default, *sendmail* is disabled.



There is no need to enable *sendmail* to send electronic mail to the network or to send and receive local Mach<sup>Ten</sup> mail among user accounts on your Mach<sup>Ten</sup> Macintosh.

### 3.2.3.3 Enable Incoming Connections

Check this box to accept connection requests for internet network services. When checked, Mach<sup>Ten</sup> will automatically start *inetd* — the internet “super server” — upon Mach<sup>Ten</sup> 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.



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 “10.0 Networking with Mach<sup>Ten</sup>”. Also detailed are instructions on how to configure PPP or SLIP interfaces (see the appropriate section “10.15.8 Using PPP” or “10.15.9 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<sup>Ten</sup>. 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<sup>Ten</sup> Macintosh directly over AppleTalk.) If a MacIP gateway and dynamic addressing are used on your network, enter an IP address of “0.0.0.0”.<sup>†</sup> (Unless you plan to use your Mach<sup>Ten</sup> networking code solely for Telnet access to some other system, it is recommended that you do not use dynamic internet addressing.)

On a system with more than one type of networking interface, the host name assigned by Mach<sup>Ten</sup> to the AppleTalk interface will be “host\_a”, where *host* is the host name selected in the General Configuration screen of the Mach<sup>Ten</sup> control panel. If this interface is the only configured interface, the host name associated with the IP address entered will simply be “host”.

---

<sup>†</sup> By default, Mach<sup>Ten</sup> scans the locally connected AppleTalk zone for an IP gateway that will provide the IP address. Mach<sup>Ten</sup> 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.

Users running EtherTalk or TokenTalk on their system may elect to configure the corresponding Ethernet or TokenRing interface under Mach<sup>Ten</sup> 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<sup>Ten</sup> 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 and share a common Ethernet cable to send and receive packets. The host name on EtherTalk assigned by Mach<sup>Ten</sup> and corresponding to the AppleTalk IP address entered would be "host\_a". 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<sup>Ten</sup> 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<sup>Ten</sup> 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<sup>Ten</sup> 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".

**Default Gateway.** This field represents the internet address in standard internet “dot” (decimal) 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 subnetting 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” format or in hexadecimal format preceded by the characters ‘0x’. The default value is taken from the MacTCP control panel. If subnetting is not employed on any of your networks, the “Netmask” field for each configured interface should be left blank. In this case, Mach<sup>Ten</sup> 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<sup>Ten</sup> allows the adjustment of the MTU for connections over SLIP lines and on TokenRing networks. For 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.

The screenshot shows the 'Networking Configuration' dialog box. At the top, there are three checkboxes: 'Enable IP Forwarding' (unchecked), 'Enable Incoming Mail' (checked), and 'Enable Incoming Connections' (checked). Below these are two columns of input fields: 'IP Addresses' and 'Netmasks'. The 'IP Addresses' column has fields for AppleTalk (at0) with value 223.255.254.81, Ethernet (ie0) with value 223.255.254.17, TokenRing (tr0) which is empty, Default Gateway with value 223.255.254.33, and Domain Name Server with value 223.255.254.1. The 'Netmasks' column has fields for AppleTalk (at0) with value 255.255.255.192, Ethernet (ie0) with value 255.255.255.192, and TokenRing (tr0) which is empty. Below these columns are two more input fields: 'SLIP MTU' with value 296 and 'TokenRing MTU' with value 4464. At the bottom of the dialog are three buttons: 'Default', 'Cancel', and 'Save'.

	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
Domain Name Server	223.255.254.1	
SLIP MTU	296	TokenRing MTU
		4464

Figure 14. Example Network Configuration



## 4.0 Launching Mach<sup>Ten</sup>

Mach<sup>Ten</sup> consists of a Macintosh application called *MachTen* and a collection of shared libraries that contains the bulk of the Mach<sup>Ten</sup> 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 15. Mach<sup>Ten</sup> Application Folder

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

In the time between starting *MachTen* and the appearance of the login prompt, Mach<sup>Ten</sup> initialization takes place (see "Figure 16. Mach<sup>Ten</sup> Initialization"). The UNIX boot process is controlled by parameterized scripts that work in conjunction with three UNIX database files — *MachTen Prefs*, *MachTen Conf* and *Install.conf* — built by the Mach<sup>Ten</sup> control panel.

The Mach<sup>Ten</sup> application's function is to load the Mach<sup>Ten</sup> 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 that reads the */etc/rc* database and executes the commands in it. With input from the */etc/rc.conf* configuration file generated from the Mach<sup>Ten</sup> control panel settings, the network interfaces are activated. Also, several server programs, or *daemons*, are started (see "Table 2. Mach<sup>Ten</sup> Daemons"). If successful in completing all of the commands in */etc/rc*, a login console is displayed.

Mach<sup>Ten</sup> daemons are described in detail in section 8 of the UNIX man pages. In Mach<sup>Ten</sup>, the startup data files ("Table 3. Mach<sup>Ten</sup> Data Files") may be configured via the Mach<sup>Ten</sup> control panel. Mach<sup>Ten</sup> data files are described in detail in section 5 of the man pages.

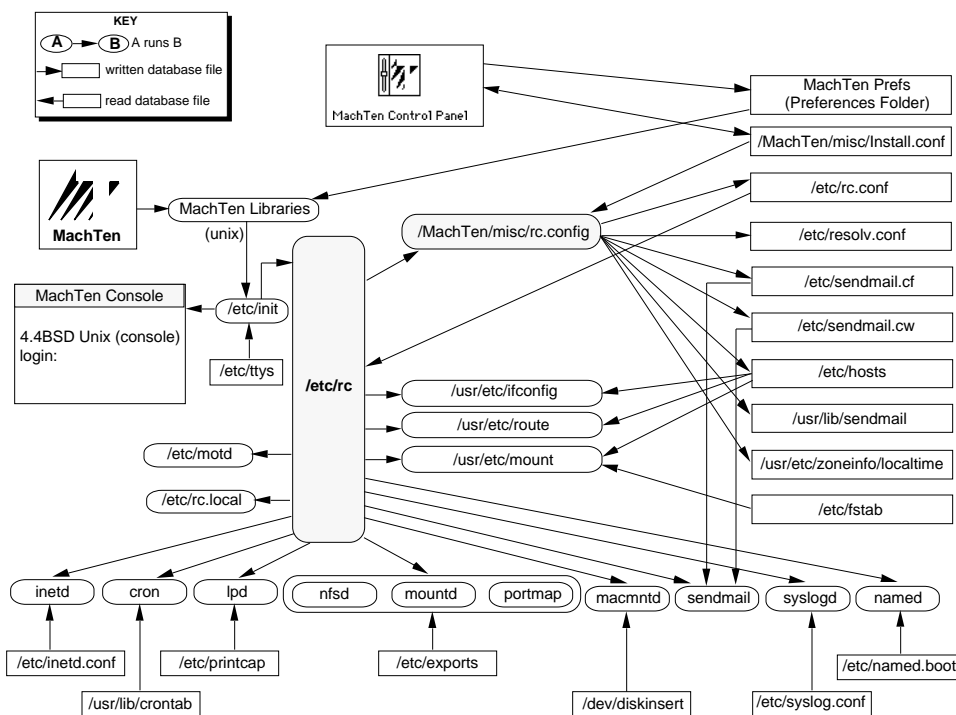


Figure 16. Mach<sup>Ten</sup> Initialization



Table 2. Mach<sup>Ten</sup> Daemons

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

Table 3. Mach<sup>Ten</sup> 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 /etc/Install.conf database built by Mach <sup>Ten</sup> 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.
/etc/localtime	Controls the time zone mappings to GMT.
/etc/rc.conf	Used by /etc/rc. Reflects parameters from the Networking Configuration and General Configuration screens of the Mach <sup>Ten</sup> control panel.
/etc/fstab	Database of file systems. Used by /sbin/mount.
/etc/sendmail.cf	Sendmail(8) configuration file.
/etc/hosts	Host name-to-internet-address mapping.

## 4.1 The Mach<sup>Ten</sup> Login Console

When Mach<sup>Ten</sup> is first started, it displays a command window<sup>†</sup> 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 17. Mach<sup>Ten</sup> Login Console”.)

Mach<sup>Ten</sup> provides three pre-configured login accounts — *root*, *mtuser* and *mtnext*. Each of these accounts has the password “MachTen”. The *root* and *mtuser* accounts default to a UNIX terminal window environment. The *mtnext* account defaults to a high-performance X desktop, using the AfterStep window manager.

In the example in “Figure 17. Mach<sup>Ten</sup> Login Console”, “fred” is the host name that was entered into the control panel when Mach<sup>Ten</sup> was installed. Log in as *root* by typing “root” (followed by a carriage return), and then typing the password “MachTen”. Observe what happens. Be careful to use lowercase. Unlike the Macintosh, Mach<sup>Ten</sup> 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<sup>Ten</sup> “terminal window” is opened (see “Figure 18. Mach<sup>Ten</sup> Terminal Window”).

After logging in, you will enter a windows environment controlled by the *wind(8)* program. Mach<sup>Ten</sup> windows let you create multiple windows, each representing a UNIX “shell” (the UNIX command interpreter). When you type commands in a Mach<sup>Ten</sup> terminal window, you are communicating with the shell. Mach<sup>Ten</sup> provides the Bourne shell (*sh(1)*) (the default), the C shell (*csh(1)*), *bash(1)* and *tcsh(1)*. Mach<sup>Ten</sup> windows are described in detail in section “4.2 Mach<sup>Ten</sup> Windows”.

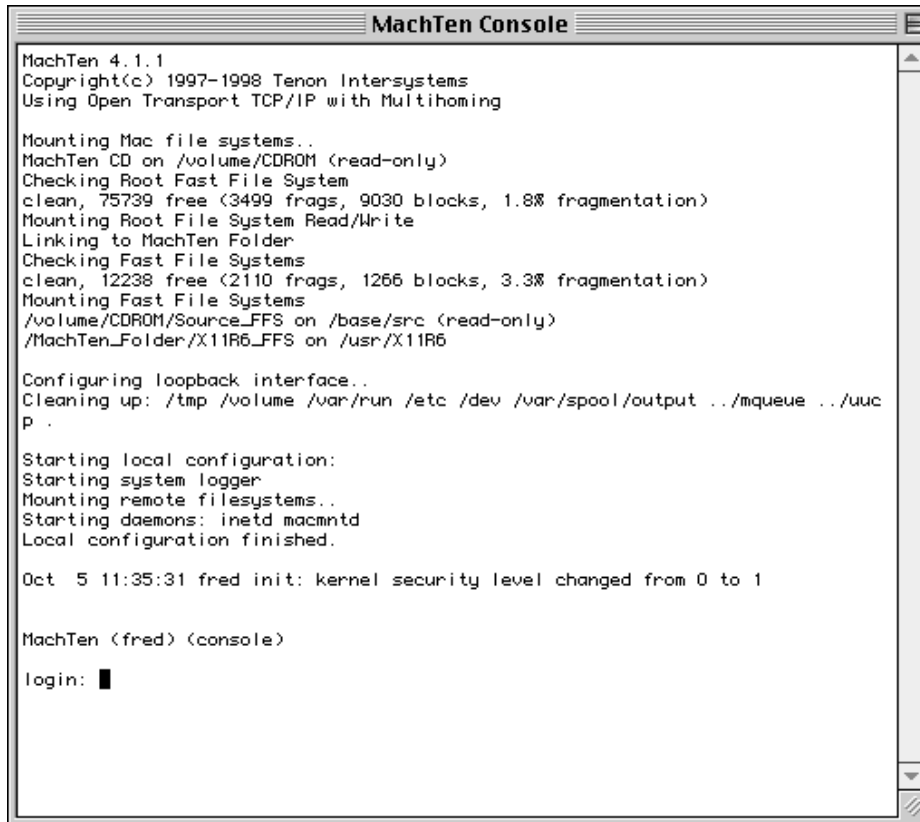
Once you log in, you should immediately invoke the *passwd(1)* program and create a password for the *root* login. This is done by typing the following:

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

For more information, refer to section “5.6.5 The Root Password”.

---

<sup>†</sup> Customizing the size and location of the console window may be achieved by using ResEdit to edit the WIND resource, ID 128, in the Mach<sup>Ten</sup> application.



```
MachTen Console
MachTen 4.1.1
Copyright(c) 1997-1998 Tenon Intersystems
Using Open Transport TCP/IP with Multihoming

Mounting Mac file systems..
MachTen CD on /volume/CDROM (read-only)
Checking Root Fast File System
clean, 75739 free (3499 frags, 9030 blocks, 1.8% fragmentation)
Mounting Root File System Read/Write
Linking to MachTen Folder
Checking Fast File Systems
clean, 12238 free (2110 frags, 1266 blocks, 3.3% fragmentation)
Mounting Fast File Systems
/volume/CDROM/Source_FFS on /base/src (read-only)
/MachTen_Folder/X11R6_FFS on /usr/X11R6

Configuring loopback interface..
Cleaning up: /tmp /volume /var/run /etc /dev /var/spool/output ../mqueue ../uuc
P .

Starting local configuration:
Starting system logger
Mounting remote filesystems..
Starting daemons: inetd macmtd
Local configuration finished.

Oct 5 11:35:31 fred init: kernel security level changed from 0 to 1

MachTen (fred) (console)
login: █
```

Figure 17. Mach<sup>Ten</sup> Login Console

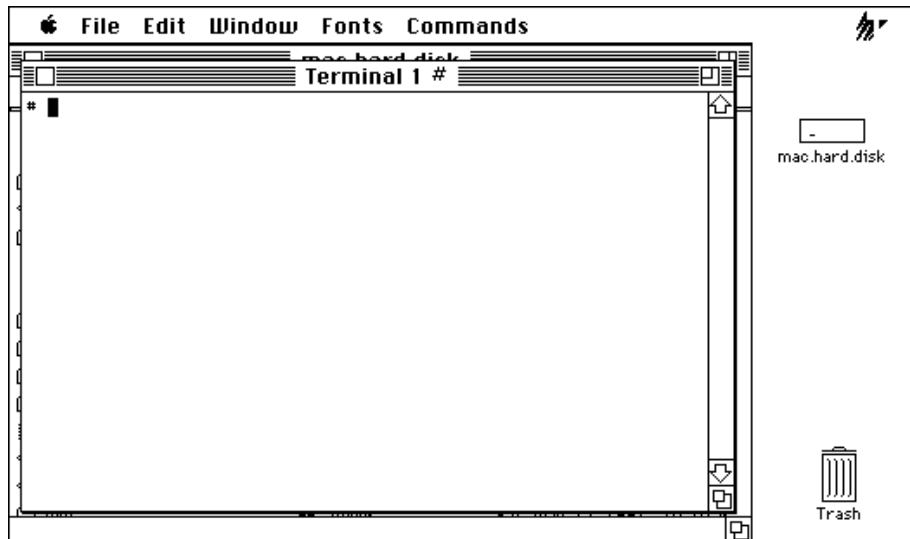


Figure 18. Mach<sup>Ten</sup> Terminal Window

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<sup>†</sup> indicating that the system is waiting for input. To establish a new password<sup>††</sup>, use the *passwd(1)* command, just as you did before as *root*.

---

<sup>†</sup> For non-*root* logins, a ‘\$’ 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.%*

<sup>††</sup> When this is done, you will have changed the password for the current user. It is now up to you to remember the password, as there is no way to “look up” a password in Mach<sup>Ten</sup>.

## 4.2 Mach<sup>Ten</sup> Windows

The Mach<sup>Ten</sup> 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<sup>Ten</sup> 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<sup>Ten</sup> 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 19. Mach<sup>Ten</sup> Windows”).

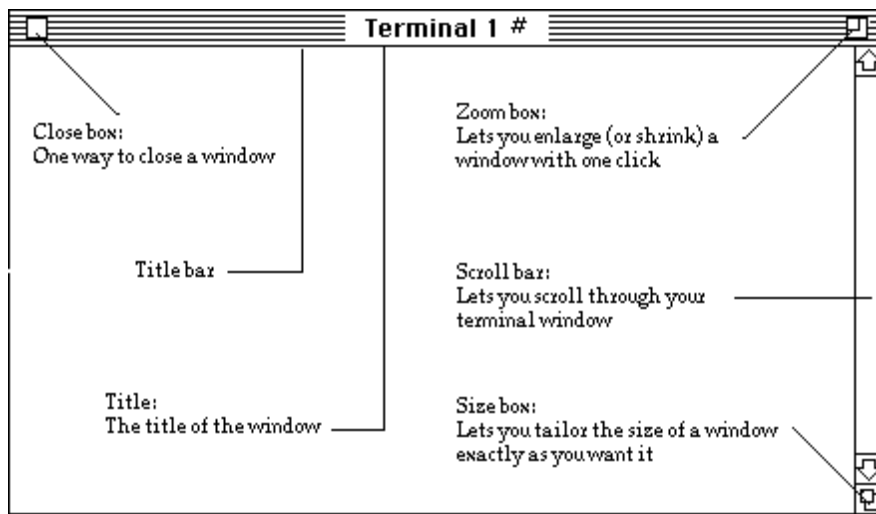


Figure 19. Mach<sup>Ten</sup> Windows

Mach<sup>Ten</sup> menus are located at the top of the screen and operate like traditional Macintosh menus.

## 4.2.1 The Apple Menu

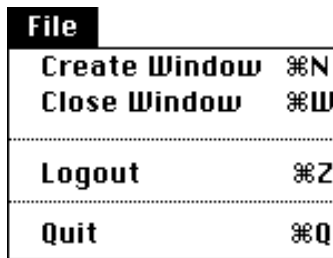
The Apple menu gives you access to “About MachTen”.



<b>About MachTen</b>	Information on how to contact Tenon.
----------------------	--------------------------------------

## 4.2.2 The File Menu

The File menu lets you create Mach<sup>Ten</sup> windows and manipulate your window environment.



<b>Create Window</b>	Command-N	Create a new terminal window.
<b>Close Window</b>	Command-W	Close the active terminal window.
<b>Logout</b>	Command-Z	Quit windows and return to the login console.
<b>Quit</b>	Command-Q	Quit Mach <sup>Ten</sup> .

### 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.

Edit	
Undo	⌘Z
Cut	⌘X
<b>Copy</b>	<b>⌘C</b>
Paste	⌘V
Clear	
<b>Select All</b>	<b>⌘A</b>
<hr/>	
<b>[Copy,] Paste, &lt;cr&gt;</b>	<b>⌘G</b>
<b>Save Selection</b>	

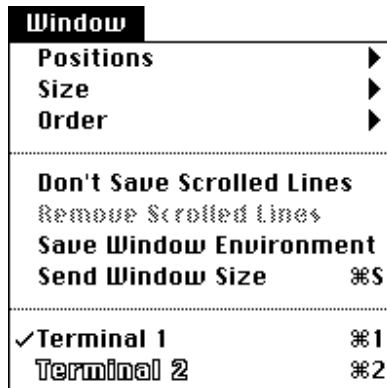
<b>Undo</b>	Command-Z	“Undo” is provided only to support desk accessories and is not active under Mach <sup>Ten</sup> .
<b>Cut</b>	Command-X	“Cut” is provided only to support desk accessories. The concept of “cutting” from a UNIX “history” screen is not meaningful. Moreover, specific applications need built-in “cut” support. In Mach <sup>Ten</sup> , traditional “cut-and-paste” functions become “copy-and-paste”.
<b>Copy</b>	Command-C	Put a copy of the selection on the Clipboard.



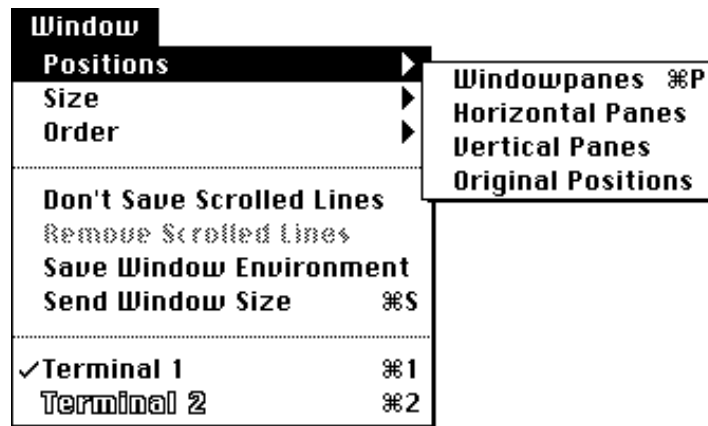
<b>Paste</b>	Command-V	Put a copy of the contents of the Clipboard into the active window. The same item may be pasted many times until the next "Copy" command.
<b>Clear</b>		"Clear" is provided only to support desk accessories and is not active under Mach <sup>Ten</sup> .
<b>Select All</b>	Command-A	Select all the text in the active window.
<b>[Copy], Paste, &lt;cr&gt;</b>	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.
<b>Save Selection</b>		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.

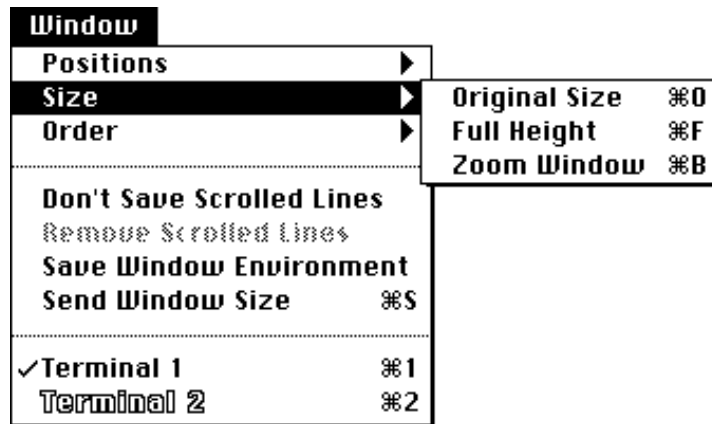


### 4.2.4.1 The Positions Sub-Menu



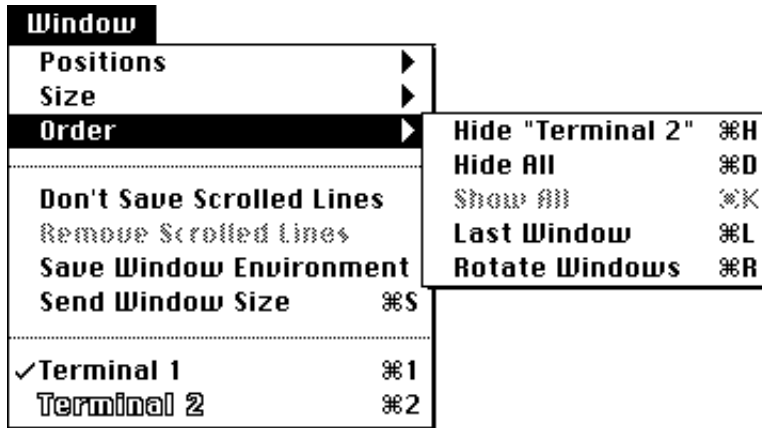
<b>Windowpanes</b>	Command-P	Position the windows so that they are all visible on the screen, creating a windowpane effect.
<b>Horizontal Panes</b>		Windowpane effect with each window as wide as the complete screen.
<b>Vertical Panes</b>		Windowpane effect with each window as tall as the complete screen.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

#### 4.2.4.2 The Size Sub-Menu



<b>Original Size</b>	Command-O	Return the active window to the size it was when it was opened.
<b>Full Height</b>	Command-F	Make the active window as tall as will fit on the screen.
<b>Zoom Window</b>	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

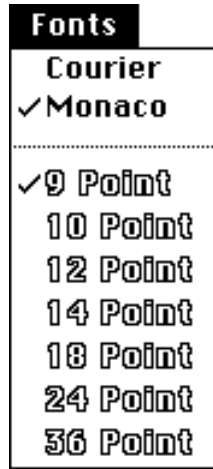


<b>Hide "Terminal #"</b>	Command-H	Make the active window invisible. A hidden window can be made visible using the "Show All" command (described below), or by selecting that window.
<b>Hide All</b>	Command-D	Make all windows invisible.
<b>Show All</b>	Command-K	Make all windows visible.
<b>Last Window</b>	Command-L	Make the previously active window the current active window.
<b>Rotate Windows</b>	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.

<b>Save Scrolled Lines /Don't Save Scrolled Lines</b>		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 says "Don't Save", and vice versa.
<b>Remove Scrolled Lines</b>		If "Save Scrolled" is enabled, this will throw away any saved lines.
<b>Save Window Environment</b>		Save a description of the sizes and locations of the current Mach <sup>Ten</sup> windows environment to a named file. Font types and sizes are also saved. This file is suitable to become the Mach <sup>Ten</sup> window environment startup file.
<b>Send Window Size</b>	Command-S	<p>Provide the string</p> <pre style="text-align: center;">stty rows n columns m</pre> <p>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.</p> <p>An example would be using <i>telnet</i> to access a Sun workstation, and trying to run a screen-oriented program such as <i>vi</i> on the remote system. After connecting to the remote system, the <i>stty(1)</i> command will inform the remote system of the terminal size to use.</p>

## 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>.



Variable-width fonts in the system will not appear in the Mach<sup>Ten</sup> Fonts menu.





## 5.0 Mach<sup>Ten</sup> Administration

This section deals with Mach<sup>Ten</sup> system initialization, user accounts, file system maintenance and virtual memory.

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

Mach<sup>Ten</sup> system administration can be performed by ordinary users. If Mach<sup>Ten</sup> systems are integrated into a corporate or campus network, the network setup tasks will no doubt be performed by the site network administrator. In an environment where there is already a UNIX administrator, Mach<sup>Ten</sup> systems can easily be incorporated into the standard UNIX environment.

## 5.1 A Word About Man Pages

Mach<sup>Ten</sup> 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<sup>Ten</sup>. The UNIX man pages are divided into numbered sections. Commands in this manual and all other Mach<sup>Ten</sup> 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<sup>Ten</sup> control panel screens are recorded in the file */etc/Install.conf*. Whenever */etc/rc* sees changes in this file, it invokes the shell script */etc/rc.config* to update several system files, including */etc/rc.conf*. */etc/rc* then reads the variables defined in */etc/rc.conf* and initializes the services and daemons specified.

Most variable definitions in *rc.conf* reflect the configuration settings entered in the Mach<sup>Ten</sup> control panel. When you modify the control panel, the changes are applied to *rc.conf* the next time Mach<sup>Ten</sup> 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 controls the configuration of the network interface(s) installed in your system:

HOSTNAME	The name of the machine running Mach <sup>Ten</sup> .
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 <i>/etc/hosts</i> 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 <sup>Ten</sup> .
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*:

CLEANUP_tmp	Remove files from <i>/tmp</i> (aka <i>/var/tmp</i> ) at startup.
MOUNT_REMOTE_FILES	Mount all file systems recorded in <i>/etc/fstab</i> ( <i>fstab</i> (5)) at startup.
PRESERVE_EDITOR_FILES	Preserve <i>ex</i> and <i>vi</i> edits in progress in the event of a system crash.
START_cron	Execute commands by date/time via <i>cron</i> (8).
START_inetd	Service incoming internet connections via <i>inetd</i> (8).
START_lpd	Manage spooled printer output via <i>lpd</i> (8).
START_macmntd	Automatically mount inserted floppies via <i>macmntd</i> (8).
START_named	Answer network queries for host names and addresses via <i>named</i> (8), the domain name server.
START_sendmail	Enable incoming internet mail via <i>sendmail</i> (8).
START_syslogd	Log system messages via <i>syslogd</i> (8). By default, this variable is set if the <i>START_sendmail</i> definition is affirmative.

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 (#).

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 <i>/etc/hosts</i> file. The <i>route(8)</i> command is invoked with this variable by <i>rc</i> 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 <i>/etc/named/boot</i> .

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 any machine-specific processes after Mach<sup>Ten</sup> 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<sup>Ten</sup> to Boot Automatically

Mach<sup>Ten</sup> 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<sup>Ten</sup> automatically, simply remove the alias files from the *Startup Items* folder.

### 5.2.2 Manual Startup

If Mach<sup>Ten</sup> has not been configured to start automatically, simply double-click the *MachTen* application in the *MachTen* folder on your Finder desktop.

## 5.3 Tailoring Mach<sup>Ten</sup> Windows

Mach<sup>Ten</sup> windows are started by the `/usr/bin/wind` daemon. Note how `/usr/bin/wind` is invoked by the `.login` file when you log on to the Mach<sup>Ten</sup> 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, a `.windrc` file must exist in your home directory. The easiest way to create a `.windrc` file is to create 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` and place it in your home directory. The changes will take effect the next time you launch Mach<sup>Ten</sup>.

It is a good idea to keep a copy of your current `.windrc` file 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 path name) 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<sup>Ten</sup>, *login(1)* creates a special variable called MACHTEN, which indicates the Mach<sup>Ten</sup> kernel version number.

To see your current environment, type:

```
printenv
```

A typical environment looks like this:

```
VIRTUAL_MEMORY=false
POIX=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:u
s=2\E[4m:ue=2\E[m:md=2\E[lm:mr=2\E[7m:mb=2\E[5m:me=2\E[m:is=\E[1;24r\E[
24;lH:if=/usr/share/tabset/vt100:rs=\E>\E[?3l\E[?4l\E[?5l\E[?7h\E[?8h:k
s=\E[?lh\E=:ke=\E[?ll\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)* under *csh(1)*, for example, type the following:

```
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.

## 5.5 Quitting Mach<sup>Ten</sup>

Selecting “Quit” from the File menu can be used to quit Mach<sup>Ten</sup> without restarting the MacOS. Mach<sup>Ten</sup> can subsequently be launched again. The command *reboot* will kill all of the UNIX processes and then quit Mach<sup>Ten</sup>. The *-q* option to *reboot* will quit Mach<sup>Ten</sup> immediately, without attempting to kill the other UNIX processes. To provide a degree of remote control (via Mach<sup>Ten</sup>) 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<sup>Ten</sup> invokes */sbin/reboot* to kill the UNIX processes and then quit Mach<sup>Ten</sup>. Control returns to the Finder to “Shut Down” or “Restart”, as appropriate.

When Mach<sup>Ten</sup> menus are used and the “Quit” item under the File menu is selected, Mach<sup>Ten</sup> invokes */sbin/reboot*. Holding down the <Shift> key while selecting this item invokes */sbin/reboot -q*. This also applies to the “hot key” combinations <Command-Q> and <Shift>-<Command-Q>.



## 5.6 Login Accounts

Since Mach<sup>Ten</sup> 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 without modifying another user's files and 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 called */etc/passwd* (see *passwd(5)*), described in sections "5.6.7 The Password File" and "5.6.8 The Group File".

Mach<sup>Ten</sup> 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<sup>Ten</sup> terminal window, you are communicating with the shell. Mach<sup>Ten</sup> 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. Both the Bourne shell and the C shell are powerful execution environments. In addition, both shells support programming languages that include flow control primitives, parameter passing, variables and string substitution. The Bourne shell 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 user 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, and you must know the *root* password.

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

If an ordinary user is also the 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.

#### 5.6.4 A Word About Security

Since Mach<sup>Ten</sup> is implemented as an application on a non-secure operating system (the MacOS), traditional UNIX-style security is not enforced by Macintosh applications. However, since Mach<sup>Ten</sup> imposes traditional UNIX permissions on files and supports traditional UNIX password protections, Mach<sup>Ten</sup> files are protected from users logging in remotely, and Mach<sup>Ten</sup> permissions are enforced on NFS exported files. Since the Mach<sup>Ten</sup> file system is simply the native Macintosh file system, traditional Macintosh security software, such as FolderBolt, works well with Mach<sup>Ten</sup> 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<sup>Ten</sup> system, the *root* account has the password “MachTen”. Once you log in, you should immediately invoke the *passwd(1)* program and change the 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 22. The */etc/group* File”).

```
%su
password: <root_password>
#                               (... whatever you need to do)
#^d                             (typing <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]
```

<b>-g (gid)</b>	Group ID of the new user.
<b>-u (uid)</b>	User ID of new user.
<b>-p (passwd)</b>	Text form of the user password; default is <i>username</i> .
<b>-n (user)</b>	Specifies user information, e.g. "John Brown"; default is <i>username</i> .
<b>-d (home)</b>	Specifies the user's home directory; default is <i>/home/username</i> .
<b>-s (login)</b>	Specifies program to run as login shell; default is <i>/bin/sh</i> (Bourne shell). The entry <i>/bin/csh</i> 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 "John" in group 10, enter:

```
adduser john -g 10 -n "John Brown"
```

*adduser* creates default *.login* and *.cshrc* files for the new user. A portion of the *.cshrc* file is shown in "Figure 20. A *.cshrc* File".

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.

---

† For non-*root* logins, a '\$' 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%*

```

.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/bin/vi'           Set Mail control variables.
  setenv SHELL /bin/csh
  setenv VISUAL /usr/bin/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 20. A .cshrc File

## 5.6.7 The Password File

Once you are 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)
- dummy password field
- 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. Since Mach<sup>Ten</sup> uses shadow passwords, the real password is stored in */etc/master.passwd*, and the password field consists of a single asterisk in the */etc/passwd* file. 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 "Figure 21. The */etc/passwd* File" notice that *root* has selected the C shell. By using the *chpass* command, you can change the *root* default shell to the Bourne shell.

```
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
ftp:*:65533:65533:Anonymous FTP:/home/ftp:/sbin/nologin
ppp:*:65532:65532:PPP Dialin Account:/tmp:/etc/pppserver
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
mta:*:10:10:Post.Office MTA:/tmp:/sbin/nologin
mtuser:*:100:20:mtuser:/home/mtuser:/bin/csh
```

Figure 21. The `/etc/passwd` 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, with no spaces)

The fields are separated by colons. The authorized user list is optional. It provides a way for users to belong to multiple groups. If the encrypted password is an asterisk, there is no password for that group.

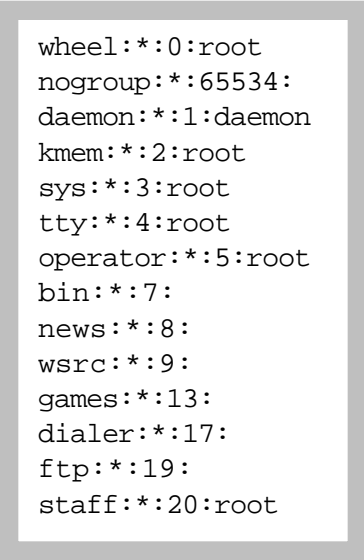
“Figure 22. The `/etc/group` File” shows how the `/etc/group` file looks when Mach<sup>Ten</sup> is first installed. This is the file used to designate group affiliations.



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 in the “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 22. 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<sup>Ten</sup>/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
ftp:*:65533:65533:Anonymous FTP:/home/ftp:/sbin/nologin
ppp:*:65532:65532:PPP Dialin Account:/tmp:/etc/pppserver
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
mta:*:10:10:Post.Office MTA:/tmp:/sbin/nologin
mtuser:*:100:20:mtuser:/home/mtuser:/bin/csh
```

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 ( <i>lpr(1)</i> ) and network communication.
sync	Special login to update write buffers. Performs a <i>sync</i> and logs out.
bin	Owner of most standard commands.
uucp	Login assigned to <i>uucp</i> requests.
news	Login used by <i>netnews</i> program. (This is a placeholder, since <i>news</i> is not implemented in Mach <sup>Ten</sup> .)

User accounts:	
root	Login for superuser. The home directory is <i>/home/root</i> and the shell is <i>/bin/csh</i> .
toor	Alternate login for superuser. The home directory is <i>/home/root</i> and the shell is <i>/bin/sh</i> .

### 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

To prevent a user from logging in, use the `passwd` 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. Removing a user involves removing the entry from the `/etc/master.passwd` file, rebuilding the password databases with the command:

```
pwd_mkdb /etc/master.passwd
```

and deleting disk files.

### 5.6.13 Changing the Message-of-the-Day

As `root` or superuser you can edit the file `/etc/motd`. If this file does not already 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<sup>Ten</sup> (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 chart in “Figure 23. Special Keyboard Characters” 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 23. Special Keyboard Characters

## 5.7 Managing Your UNIX Environment

### 5.7.1 What's Running?

Since Mach<sup>Ten</sup> 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 Macintosh and UNIX program that is running.

“Figure 24. Using `ps` to Show What is Running” shows the sample output from a `ps -lax` command.

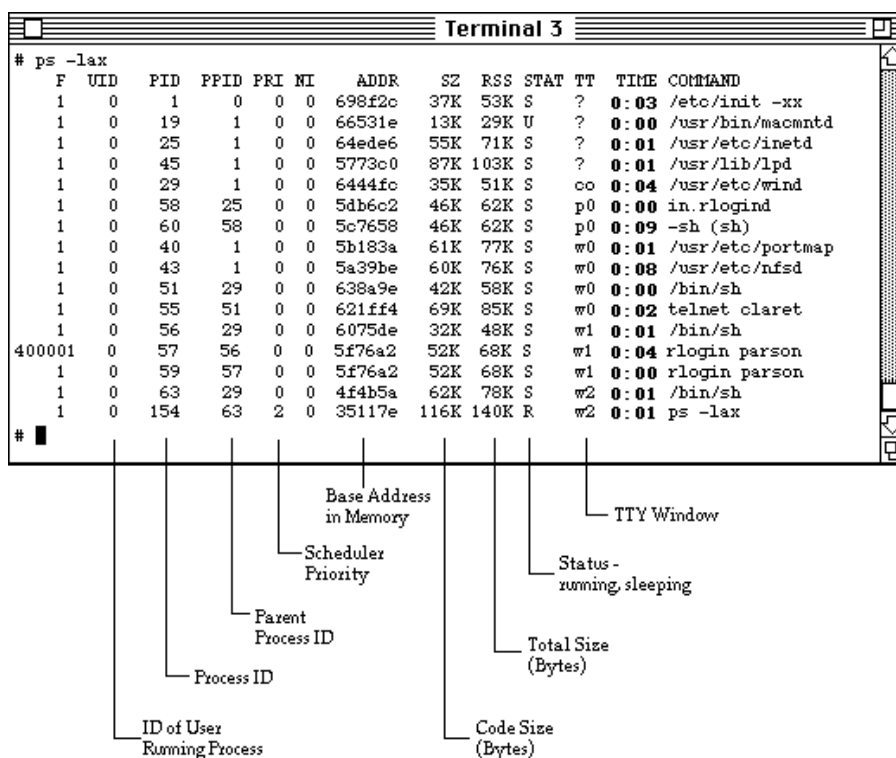


Figure 24. Using `ps` to Show What is Running

## 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 (PID) 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 24. Using *ps* to Show What is Running”.

## 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. 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 &
```

## 5.7.4 Shell Files

Mach<sup>Ten</sup> provides a means for executing multiple commands by defining them within a file. The file can then be run as a single command. Use an 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	run the <i>date</i> command
who	run the <i>who</i> command
pwd	run the <i>pwd</i> command
w startup	write to a file named <i>startup</i>
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<sup>Ten</sup> File Systems

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<sup>Ten</sup> augments the MacOS HFS or HFS+ file system with two UNIX file system implementations — a native fast file system (FFS) and a UNIX file system (UFS) that sits on top of the MacOS HFS or HFS+.

### 6.0.1 Fast File System (FFS)

The FFS implements a traditional native UNIX fast file system within a single Macintosh file. There is no need to reformat or partition a volume to create an FFS within a file. For an FFS within a file, Mach<sup>Ten</sup> continues to use the services of the Macintosh File Manager, but only for basic I/O operations. Macintosh files may be stored in an FFS, but they are not available for access by Macintosh applications.

### 6.0.2 UNIX File System (UFS)

The UNIX file system is implemented on top of the Macintosh hierarchical file system. Mach<sup>Ten</sup> imposes UNIX file system semantics on the Macintosh by mapping the UNIX file system requests onto the appropriate Macintosh File Manager routines. UNIX files are stored alongside Macintosh files, so there is no requirement to reformat/partition existing HFS volumes.

## 6.1 UNIX Fast File System Overview

Mach<sup>Ten</sup>'s FFS is derived from the Berkeley BSD UNIX high-performance file system. It is a simple and elegant file system that has its roots in the original UNIX file system developed at Bell Labs. Detailed information may be found in two supplementary documents — “A Fast File System for UNIX” and “Fsck - The UNIX File System Check Program”. These are located in the *Documentation* folder under */MachTen UNIX Docs/Sys Admin's Docs (smm)/fastfs.pdf* and *fsck.pdf*.

### 6.1.1 File System Organization

UNIX disks are divided into one or more partitions. Each partition may contain one file system, and a file system never spans multiple partitions. Critical file system sizing parameters are stored in a “super-block”, which is replicated to protect against catastrophic loss.

The file system stores files. Certain files are designated as directories that contain pointers to other files, some of which may be other directory files. Every file has an associated descriptor called an “inode”, which contains ownership, permissions, time stamps and pointers to assigned data blocks.

A disk partition is divided into one or more areas called “cylinder groups”. A small percentage of each cylinder group is taken for some bookkeeping information, including a redundant copy of the super-block, inode slots, a bitmap of available data blocks, and data block usage summary information.

A static number of inodes is allocated when the file system is created. The default policy is to create one inode per 2048 bytes of space in the cylinder group, which is expected to be far more than will be needed.

UNIX files are a single stream of bytes, with no operating system-imposed format. Text files use the ASCII “LF” character (“\n”) to denote an end-of-line. Binary files often begin with a four-byte code identifying the format of the data that follows.

## 6.1.2 File Names

UNIX file and directory names are limited to 255 characters. MacOS file names are limited to 31 characters. Names may not include the “/” character because it is used in path names to separate directory and file name components. An absolute path name begins with a “/”; anything else is relative to the current working directory.

Name matching is case sensitive. (MacOS file names are not case sensitive; however, Tenon's extension enables case sensitivity in Tenon's UFS file system.)

## 6.1.3 Access Permissions

UNIX file and directory access rights are divided into read, write and execute permissions. “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 and scripts allows a user to execute the file as a program, and for directories allows a user to search the directory. Each file and directory is tagged with a user and group ID. Three levels of permissions are specified — one for the owner of a file, a second for the members of the designated group, and a third for “everyone else”.

## 6.1.4 Time Stamps

UNIX saves three time stamps for each file — the time the file's data was modified, the time the file's attributes were last modified, and the time the file was last accessed (i.e., the data was read).

### 6.1.5 Link Counts

Physical files are uniquely identified by their inode, but they may be identified by multiple names. The link count is simply the number of references to an inode.

### 6.1.6 Hard Links

The mapping of a file name in a directory to an inode is known as a hard link. Hard links are not allowed to cross file system boundaries.

### 6.1.7 Symbolic Links

Symbolic links are special files whose content is the path name of the intended file or directory. The path name may be absolute or relative, and may refer to other file systems. Symbolic links may point to other symbolic links.

## 6.2 Macintosh Hierarchical File System Overview

The MacOS method of organizing files is known as the hierarchical file system (HFS). Detailed information about the Macintosh file system and File Manager may be found in the "Files" manual, which is part of the "Inside Macintosh" documentation series published by Addison-Wesley.

### 6.2.1 File System Organization

Each Macintosh hard disk is formatted into a number of 512-byte addressable units known as "logical blocks". A "volume" is a consecutive sequence of these blocks. Small disks are typically used whole as a single volume, while large disks are often partitioned into two or more volumes.

#### 6.2.1.1 Volumes and Folders

A volume consists of overhead, file storage and free space. The overhead includes boot blocks, bitmap, master directory blocks, catalog and extents files. The catalog file maintains the hierarchy of directories and files on a volume, and the extents overflow file tracks assignment of "allocation blocks" to files which cannot be stored in the catalog file.

File storage is assigned in units of allocation blocks, which are one or more consecutive logical blocks. A volume has at most 65,536 allocation blocks. This addressing constraint dictates the ratio of logical blocks to an allocation block. Files are stored in directories (also known as folders), which are themselves stored in directories. Each directory and file is assigned an integer ID which uniquely identifies it on a given volume.

Every volume has a top directory known as the "root" directory, with a directory ID of 2. When a volume is mounted for use, the MacOS assigns a volume reference number that remains valid as long as the volume is mounted. Macintosh files are uniquely specified with a volume reference number, a parent directory ID and a file name.

### 6.2.1.2 File Contents

A Macintosh file has two forks — a “data fork” and a “resource fork”. File data resides in the data fork, while file resources are stored in the resource fork. An application would typically have resources such as menus, dialog boxes, icons, and even code segments. A document file could have resources such as preference settings, window locations, fonts and icons. Files are marked with a four-byte creator and type, usually shown as alphanumeric tags, such as “APPL”, “TEXT” or “BINA”. The creator identifies the application that created the file, and the type generally indicates the file content. Text files use the ASCII “CR” character (“\r”) to denote an end-of-line.

### 6.2.2 File Names

Macintosh file and folder names are limited to 31 characters. Names may not include the “:” character because it is used in path names to separate volume, directory, and the file name components. An absolute path name begins with a volume name. A path name relative to the current working directory begins with a “:”. The File Manager is not case sensitive when matching file names; however, it does not ignore diacritical marks. (MacOS file names are not case sensitive; however Tenon’s extension enables case sensitivity in Tenon’s UFS file system.)

### 6.2.3 Access Permissions

Permissions for files on local volumes are based on the order and manner in which programs open them. Files can be opened for reading, writing or exclusive read/write. Write access can be denied on an individual file basis by locking it via the Finder’s “Get Info” dialog box. The File Manager also prevents locked files in the Trash from being deleted on an “Empty Trash” request. Setting a volume to read-only (by hardware or software) prevents any changes to all files on that volume.

## 6.2.4 Time Stamps

The Macintosh File Manager saves three time stamps for each file — the time the file was created, the time the file was last modified, and the time the file was last backed up (rarely used in the Macintosh world).

## 6.2.5 Aliases

An alias is a special kind of file that represents a file, folder or volume.

## 6.3 Mach<sup>Ten</sup> FFS

Mach<sup>Ten</sup>'s implementation of the UNIX Fast File System (FFS) allows fast file systems to be stored within Macintosh HFS files. Storing a fast file system in a file allows existing volumes to be used, without the need for additional partitioning, provided they have sufficient space to store the fast file system.

### 6.3.1 FFS Within a File

Macintosh files containing an FFS are treated as block devices by Mach<sup>Ten</sup>. They are set with creator "MUMM" and type "BLK". These files are visible to Finder and may be moved or copied to other folders or volumes. They may also be dragged to the Trash for removal. Be sure that this is intended before requesting "Empty Trash". These files should never be selected for write access by other Macintosh applications. Even minor changes by other Macintosh applications could render the fast file system file unusable by Mach<sup>Ten</sup>. See section "6.9.1.1 Creating an FFS Within a File".

## 6.4 Mach<sup>Ten</sup> UFS

Mach<sup>Ten</sup>'s implementation of the local UNIX file system (UFS) uses the Macintosh File Manager, enabling Mach<sup>Ten</sup>'s UNIX files to be stored alongside Macintosh files. However, there are some constraints imposed by the Macintosh File Manager that the astute user should be aware of when porting other UNIX applications to Mach<sup>Ten</sup>.

### 6.4.1 File Names

#### 6.4.1.1 Maximum Number of Characters

The Macintosh File Manager allows a maximum of 31 characters in a file name; hence, Mach<sup>Ten</sup> UFS file names are limited to 31 characters.

#### 6.4.1.2 Case-Sensitive File Names

Mach<sup>Ten</sup> supports case-sensitive file names by appending a four-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<sup>Ten</sup> 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 27 characters or less in length. Attempting to create a new file that collides with an existing case-insensitive file name of greater than 27 characters will fail.



### 6.4.1.3 Component Separators

The Macintosh File Manager uses the character “:” to separate names into volume, folder and file name components. Thus, Macintosh file names cannot contain the “:” character. UNIX uses the character “/” for the same purpose. Mach<sup>Ten</sup> 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.

### 6.4.1.4 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.<sup>†</sup> Non-printable characters are translated to a percent sign (“%”) followed by a two-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 <sup>Ten</sup>
Filename™	Filename%aa
Filename®	Filename%a8
Filename©	Filename%a9

---

<sup>†</sup> For more information on AppleSingle, refer to section “6.7.1 AppleSingle Encapsulation”.

## 6.4.2 Linked Files

In UNIX, linking allows several file names to be associated with the same physical file. Mach<sup>Ten</sup> properly supports UNIX hard and soft links via the *link(2)* and *symlink(2)* system calls.

### 6.4.2.1 Hard Links

Mach<sup>Ten</sup> 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<sup>Ten</sup>'s hard links from one volume to another will not work. Therefore, you cannot use the Finder (or any other Macintosh tool) to copy or move Mach<sup>Ten</sup> hard links from one Macintosh volume to another. This restriction is similar to the traditional UNIX restriction — no cross-disk hard links.

Mach<sup>Ten</sup>'s 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'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 UFS files.

### 6.4.3 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<sup>Ten</sup> does not attempt to fabricate a correct directory link count for UNIX directories on local Macintosh file systems<sup>†</sup>.

### 6.4.4 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<sup>Ten</sup> 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<sup>Ten</sup> can modify or remove the file.

### 6.4.5 File Types

The Macintosh File Manager saves a file type and file creator as part of the attributes for every file. Mach<sup>Ten</sup> uses the identifier “MUMM” for UNIX files. In Mach<sup>Ten</sup> the term “UNIX file” is used to refer to any file with a creator “MUMM”; the term “Macintosh file” is used to refer to all other files. Note that Macintosh applications can see and often access the UNIX files, since they are simply Macintosh files with a creator “MUMM”. For example, you can drag-and-drop a “MUMM/TEXT” Mach<sup>Ten</sup> file over Netscape, although this will not work with a “MUMM/BINA” file (a UNIX executable).

Mach<sup>Ten</sup> makes the distinction between UNIX files and Macintosh files.

---

<sup>†</sup> Mach<sup>Ten</sup> always returns a link count of 2 for UFS directories.

All Mach<sup>Ten</sup> documents and files are owned by the application named *MachTen*. Mach<sup>Ten</sup> 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.

### 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<sup>Ten</sup>. This is the simplest, fastest and most common type of file. When Mach<sup>Ten</sup> 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<sup>Ten</sup> 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<sup>Ten</sup> 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<sup>Ten</sup>. For more information on the handling of text files in Mach<sup>Ten</sup>, refer to section "6.8 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<sup>Ten</sup> — one for the compile-time Header Call definitions, and the other for the run-time dynamic linking.

## 6.4.6 File Permissions

Mach<sup>Ten</sup> stores the user ID, group ID and mode bits in a reserved area of the resource fork for each UNIX file. Mach<sup>Ten</sup> can also set these attributes for Macintosh files.<sup>†</sup> Note that this protection only applies to file access from Mach<sup>Ten</sup>; it does not apply to the Finder or other Macintosh applications.

On Macintosh HFS file systems, Mach<sup>Ten</sup> 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 25. Example of File Permissions

<sup>†</sup> The Mach<sup>Ten</sup> application has resources which allow the user to configure the default UNIX access privilege modes for Macintosh files and folders (see “Appendix B, Resources — Mach<sup>Ten</sup> Application”, in the online Power Mach<sup>Ten</sup> User's Guide).

## 6.4.7 Time Stamps

When Mach<sup>Ten</sup> 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 last-backed-up time is used for the UNIX last-accessed-time. When Mach<sup>Ten</sup> modifies the 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. The file's Macintosh modification time is set to the UNIX attribute modification 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" time stamp, the Mach<sup>Ten</sup> interpretation of the Macintosh time stamps 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" time stamp. A Macintosh application that 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" time stamps, which will be interpreted by UNIX programs in the proper "the-data-(or attribute)-has-been-modified" sense. In some cases, the Macintosh editor may only change the UNIX attribute modification time. In this case, it may be necessary to *touch(8)* the file from within Mach<sup>Ten</sup> to rebuild the project correctly.



## 6.5 Mach<sup>Ten</sup> Root File System Layout

The Mach<sup>Ten</sup> root file system is delivered as an FFS within a file. The FFS file is named *MachTen\_FFS*, and resides in the same folder as the Mach<sup>Ten</sup> application. A companion folder named *MachTen\_HFS* provides for UFS-oriented storage.

At the top of the Mach<sup>Ten</sup> file system hierarchy is the UNIX *root* directory, also referred to as */*. The path *//* is a special case in Mach<sup>Ten</sup> that allows UNIX processes to refer to files and folders outside of the Mach<sup>Ten</sup> root. *//* is interpreted to mean the real root of the Macintosh volume that contains the Mach<sup>Ten</sup> file system hierarchy. The corresponding routines in the Mach<sup>Ten</sup> libraries (*getwd(3)*) have been modified to understand a path name that begins with *//*.

### 6.5.1 The root Directory Tree

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

Name	Description
CDROM	Mount point for Mach <sup>Ten</sup> CD-ROM
base	Base of source code and place for building binaries in Mach <sup>Ten</sup> distribution
bin	Basic user utilities
bootvol	Symbolic link or mount point for the root of the MacOS boot volume
dev	Block, character and other special device files
etc	System configuration files and scripts
hfs	Symbolic link to <i>MachTen_HFS</i>
lost+found	Storage for files lost or corrupted by file system damage
mnt	Temporary mount point
sbin	Basic system administration utilities

Name	Description
tmp	Directory for temporary files
usr	Contains majority of system utilities and files
var	Multi-purpose log, temporary, transient and spool files
volume	<i>macmntd</i> creates subdirectory mount points for removable volumes

## 6.5.2 The usr Directory Tree

Name	Description
X11	Symbolic link to Mach <sup>Ten</sup> X Window software (X11R6)
X11R6	Mach <sup>Ten</sup> X11R6 software
bin	Binaries
doc	Miscellaneous documentation
include	C and C++ header files
info	Documentation for GNU programs
lib	Compile-time libraries
libexec	Catchall for system daemons
local	Mount point for file system containing local applications and support files
macppc	PowerPC-specific files go here
man	<i>troff</i> source for man pages
sbin	System administrator binary files
share	Text and database files readily shared among 4.4BSD systems
src	<i>rpm</i> support tree

### 6.5.3 The var Directory Tree

Name	Description
adm	Administrative files
at	Timed command scheduling files
backups	Miscellaneous backup files
cron	<i>cron</i> data files
db	Database files
games	Miscellaneous game status and log files
lib	Dynamic, machine-specific application support files
lock	Storage for <i>emacs</i> file locking
log	Miscellaneous system log files
msgs	System messages
obj	Object files
preserve	Temporary home of files preserved when editors fail
run	System information files; rebuilt after each reboot
spool	Miscellaneous printer spooling directories
tmp	Temporary files that are not discarded between system reboots

## 6.5.4 Major System Administration Files

For details about these files, see online man pages.

Name	Description
/etc/fstab	Contains file system mount information ( <i>fstab(5)</i> )
/etc/group	Defines mapping of group numbers to names and authorized users ( <i>group(5)</i> )
/etc/hosts	Defines host names, aliases and IP address
/etc/master.passwd	Defines the set of users authorized to log in. Provides mapping between user ID and name, default group, home directory and shell ( <i>passwd(5)</i> )
/etc/printcap	Contains the list of available printers and the parameters used by the line printer daemon ( <i>printcap(5)</i> )
/etc/rc	Contains the script of commands <i>init(8)</i> is to execute to establish the desired environment ( <i>rc(8)</i> )
/etc/rc.conf	Contains variables derived from information entered in the Mach <sup>Ten</sup> control panel for use by <i>/etc/rc</i> when starting Mach <sup>Ten</sup>
/etc/syslog.conf	<i>/usr/sbin/syslogd</i> configuration file ( <i>syslog(8)</i> )
/etc/ttys	Specifies terminal support information for <i>init(8)</i> ( <i>ttys(5)</i> )
/usr/share/misc/termcap	Defines the capabilities for numerous terminal types ( <i>termcap(5)</i> ). Being phased out by <i>ncurses</i>

## 6.6 Mounting Macintosh Volumes

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

In Mach<sup>Ten</sup>, the *mount(8)* command has been modified to accept a Macintosh volume name, in addition to a UNIX device name, for mounting local file systems. When Mach<sup>Ten</sup> is started, the *root* folder of the volume from which Mach<sup>Ten</sup> was launched is automatically mounted as `"/"`.

Once Mach<sup>Ten</sup> 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 `/mnt/Untitled`, the following commands may be used:

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

Note that the path where the volume is to be mounted must exist (`/mnt` is created by the Mach<sup>Ten</sup> installation program), and that volume names with spaces must be contained within quotes.

UNIX provides a database file called `/etc/fstab` to specify the path on which each file system is to be mounted. In Mach<sup>Ten</sup>, the format of this file has been extended to accept Macintosh volume names, just like the *mount* command (see *fstab(5)*).

### 6.6.1 Mounting Permissions

Since Mach<sup>Ten</sup> is targeted for a single user, non-networked system, the *mount* and *umount* commands have been set to run as *root*. This allows you to log in as a normal user and still be able to mount or unmount file systems without having to be superuser to gain temporary *root* privileges.

## 6.6.2 Automatic Mounting of Removable Media

Mach<sup>Ten</sup> provides a special UNIX daemon called *macmntd(8)* to handle automatic mounting of removable media like floppies, CDs, and Bernoulli, Jaz, Syquest and Zip 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 directory named */volume/<volume name>* and mount the volume on this directory.

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).

## 6.6.3 Unmounting Macintosh Volumes

In Mach<sup>Ten</sup>, 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)
```

## 6.6.4 Automatic Unmounting

In Mach<sup>Ten</sup>, Macintosh volumes will be automatically unmounted when the Finder is used to drag the disk to the Trash.

## 6.6.5 Formatting Floppies

In UNIX, it is often desirable to write directly to a device (disk) without regard for file system structures. Mach<sup>Ten</sup> permits this type of raw access. Under Mach<sup>Ten</sup> 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<sup>Ten</sup> 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.

Mach<sup>Ten</sup> 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.

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

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

## 6.7 Accessing Macintosh Files from UNIX Applications

### 6.7.1 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<sup>Ten</sup> uses this method of encapsulation.

### 6.7.2 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<sup>Ten</sup> 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<sup>Ten</sup> when the file is accessed by UNIX applications. Conversely, when an AppleSingle format file is written by a UNIX application such as *tar(1)*, Mach<sup>Ten</sup> automatically re-creates a proper Macintosh file with the specified data and/or resource forks.



## 6.7.3 Utilities for Manipulating Macintosh Files

### 6.7.3.1 `dfork` and `rfork`

Mach<sup>Ten</sup> provides a UNIX filter called *dfork(1)* which reads an AppleSingle format file from its standard input and writes only the data fork to the standard output. Mach<sup>Ten</sup> also provides a filter called *rfork(1)* which reads an AppleSingle format file from its standard input and writes only the resource fork to the standard output.

### 6.7.3.2 `finderinfo`

Mach<sup>Ten</sup> provides a UNIX utility called *finderinfo(1)* for accessing and changing the Finder information of a Macintosh file, such as creator and type. Options to *finderinfo* specify which Finder information fields to change. The following command illustrates changing the creator to "MUMM" and type to "TEXT" for a Macintosh file:

```
finderinfo -C MUMM -T TEXT <Macintosh_file>
```

Note that using this utility in no way changes the data in either fork of a file.

### 6.7.3.3 `restool`

Mach<sup>Ten</sup> provides a UNIX utility called *restool(1)* for merging the resources of one (or more) Macintosh file(s) into another. *restool* requires the services of the Macintosh Resource Manager, and thus can only work with UFS files.

## 6.8 Text File Manipulation

Mach<sup>Ten</sup> gives you the capability to integrate UNIX-created ASCII text files into documents that have been created by Macintosh word processor programs (such as Microsoft Word) and to export Macintosh-created text to UNIX applications.

### 6.8.1 Alternating Between Macintosh and UNIX Text

Since the Macintosh and UNIX text file formats differ in the choice of line terminator, in some cases it may be necessary to translate the data from one format to the other. Mach<sup>Ten</sup> provides several mechanisms to convert text files from the Macintosh format to the UNIX format.

Mach<sup>Ten</sup> 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<sup>Ten</sup> automatically translates "MUMM/TEXT" files to the UNIX text file format when they are accessed by UNIX applications.

#### 6.8.1.1 mactext

The *mactext(1)* utility program will create "MUMM/TEXT" files from traditional UNIX files (type "BINA", "n1" termination). The resultant file will be recognizable by UNIX and Macintosh applications.

### 6.8.1.2 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<sup>Ten</sup> UNIX system for access by the remote system.

### 6.8.1.3 Unix <-> Text

The *Unix <-> Text* application 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<sup>Ten</sup> common text 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.

### 6.8.1.4 dfork.text

Mach<sup>Ten</sup> provides the UNIX utility program *dfork.text(1)* to extract the data fork from its standard input (typically a Macintosh text file), translate the "cr" character to "nl", and produce the results on its standard output. This utility is useful if you want to leave an 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>
```

## 6.8.2 Editing Tools

### 6.8.2.1 UNIX Editors

Mach<sup>Ten</sup> provides a number of UNIX text editing programs, such as *vi(1)*, *ex(1)* and *ed(1)*. Another popular UNIX editor is GNU *emacs*. UNIX text editing programs will operate correctly on all Mach<sup>Ten</sup> text files of type "BINA", as well as files in the Mach<sup>Ten</sup> common text format ("MUMM/TEXT"). They will not recognize Macintosh "TEXT" files due to the line termination differences previously described.

### 6.8.2.2 Macintosh Editing Applications

A few Macintosh editing applications may be found on the Mach<sup>Ten</sup> CD-ROM in the sub-folder *Utilities: Text Utilities*. Please be sure to register any shareware that you decide to use on an ongoing basis.

## 6.8.3 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.

## 6.9 File System Administration

### 6.9.1 Creating File Systems

To create an FFS, you must decide how large you want it and where to place it on your disks. When sizing the file system, keep in mind that about 4% of space goes to overhead, and that (by default) 10% is held as a reserve. When free space falls below the reserve, new files may be opened only by *root*. A different reserve percentage may be set during file system creation.

Since UFS file systems are simply Macintosh HFS volumes viewed through a UNIX lens, there are no Mach<sup>Ten</sup> activities required other than creating the directories on which to mount them.

#### 6.9.1.1 Creating an FFS Within a File

Mach<sup>Ten</sup> includes an installer utility to greatly simplify the task of adding fast file systems to your Mach<sup>Ten</sup> system. *FFS\_Installer* is an installer program for creating additional fast file systems for Mach<sup>Ten</sup>.

To create a new FFS, launch this program and select the desired size FFS. Install it into the *MachTen* folder and launch Mach<sup>Ten</sup>. This new FFS will be mounted automatically when Mach<sup>Ten</sup> starts up.

To create a custom FFS within a file, first decide on the amount of storage you need. Identify the HFS volume that will accommodate it and, if necessary, mount it for access under Mach<sup>Ten</sup>. (See section “6.9.3 Mounting File Systems” for more information on mounting.) Then use the *newfsf(8)* command as follows:

```
newfsf -s <size> <FSfile_pathname>
```

where *size* is the number of 512 byte blocks that provide the desired storage, and *pathname* specifies the name of the FFS file to create.

Following creation, an FFS within a file must be mounted under Mach<sup>Ten</sup> before it can be used. If necessary, create the directory on which you will mount it. The *mount* command syntax is as follows:

```
/sbin/mount -t ffs <FSfile_pathname> <directory>
```

If you plan on using this file system on a regular basis, consider making an entry in */etc/fstab* so that it is automatically mounted on each launch of Mach<sup>Ten</sup>.

## 6.9.2 Checking/Repairing File Systems

A chief FFS maintenance activity is the periodic checking and repairing of FFS bookkeeping structures using the *fsck(8)* utility. During startup, Mach<sup>Ten</sup> automatically checks each FFS registered in */etc/fstab* unless the file */clean\_start* exists. This file is only present if Mach<sup>Ten</sup> was cleanly shut down on the prior launch.

The Macintosh File Manager assumes responsibility for checking and making minor repairs to Macintosh HFS volumes. Other Macintosh applications, such as *Disk First Aid*, perform more extensive checks and repairs. Therefore, Mach<sup>Ten</sup> takes no additional action to check or repair UFS file systems.

### 6.9.2.1 fsck

Whenever the state of an FFS is in doubt, run *fsck(8)* to check and correct the consistency of the FFS bookkeeping information. The following command illustrates how to check an FFS within a file:

```
/sbin/fsck <FSfile_pathname>
```

### 6.9.2.2 Fast File First Aid

The *fck* utility is also available as a standalone Macintosh application *Fast File First Aid*, located on the Mach<sup>Ten</sup> CD-ROM in the sub-folder *Utilities: Fast File Utilities*. *Fast File First Aid* should be used when Mach<sup>Ten</sup> is unable to boot due to root file system damage. It performs a more aggressive and interactive repair of specified FFS files. The document *fck.pdf*, located on the Mach<sup>Ten</sup> CD-ROM in the sub-folder *Documentation: MachTen UNIX Docs: SysAdmin's Docs(smm)*, contains some information that may be useful in this process.

### 6.9.3 Mounting File Systems

File systems must be mounted before they may be accessed under Mach<sup>Ten</sup>. During startup, the Mach<sup>Ten</sup> kernel automatically mounts the *root* file system (“/”) and the *root* folder of the Macintosh volume on which Mach<sup>Ten</sup> resides (“//”). The */etc/rc* script optionally mounts file systems registered in */etc/fstab*. If Mach<sup>Ten</sup> is not running on the boot volume, the boot volume will be mounted on */bootvol* by the */etc/rc* script.

#### 6.9.3.1 Mounting an FFS Within a File

Use the following command to mount an FFS within a file:

```
/sbin/mount -t ffs <FSfile_pathname> <directory>
```

If the FFS resides on a different Macintosh volume, refer to section “6.9.3.2 Mounting a Macintosh Volume”.

### 6.9.3.2 Mounting a Macintosh Volume

Use the following command to mount a Macintosh volume:

```
/sbin/mount <Mac Volume> <directory>
```

### 6.9.4 Unmounting File Systems

Unmounting FFS and UFS file systems is done with the *umount* command. Its argument may be a mount point path name, an FFS file name, or a Macintosh volume name. Any argument that contains special characters, such as a space, should be inside double quotes.

```
/sbin/umount /mnt  
/sbin/umount <FSfile_pathname>  
/sbin/umount /dev/sd<#P>  
/sbin/umount <Mac Volume>
```

### 6.9.5 Removing File Systems

When an FFS is no longer needed, its space should be returned for other purposes. The first action is to unmount it. The second is to remove any entry in */etc/fstab* that automatically mounts it. The third is to remove it, as described in section “6.9.5.1 Removing an FFS Within a File”.

#### 6.9.5.1 Removing an FFS Within a File

Delete an FFS file with Finder, or by using the following command:

```
rm -f <FSfile_pathname>
```



## 6.9.6 Space Management

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. With respect to the file system, a key administrative role is to keep the size (and number) of temporary UNIX working files 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.

You will monitor your own usage on a single user system.

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
/var/log/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 time stamp a zero length file.

## 6.9.7 Backing Up and Archiving File Systems

Backing up and archiving are two methods of file system protection. Backing up involves making a complete copy of a file system onto another hard disk or a removable medium such as tape, Jaz, Zip, Bernoulli or Syquest. Archiving usually involves storage of specific files on similar media as for backups, or even Macintosh floppies.

Archiving may be done for preserving development milestones or for recapturing space consumed by large, infrequently used files that can be restored when required. Backups should be done on a regular basis to prevent loss due to disk failure or human error. Archiving should be done as needed.

There are three programs that are useful for backing up and archiving — *tar(1)*, *dump(1)* and *restore(8)*. *tar* may be used to both write and read backups or archives. It writes a portable, structured image that is independent of the source file system. *dump* and *restore* may be used for complete and incremental backups of FFS file systems. They are file system structure dependent.

### 6.9.7.1 Tape Devices

Mach<sup>Ten</sup> supports DAT (4mm) and Exabyte (8mm) tape drives. The Mach<sup>Ten</sup> 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 (e.g., ClubMac and Wang DAT 2600)
Exabyte	Model EXB-8200

Mach<sup>Ten</sup> 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<sup>Ten</sup> uses four device files for each tape drive. For example, the following device files apply to the first tape drive, unit 0. Mach<sup>Ten</sup> uses a scan of the SCSI chain (from ID-6 to ID-0) to identify tape sub-systems 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).

### 6.9.7.2 tar

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 both FFS and local Macintosh volumes. Macintosh files will be written in AppleSingle format to preserve both the resource and data forks of the file.

An additional utility called *mt* is provided with Mach<sup>Ten</sup> to position, rewind and offload tapes. It may also be used to print out tape information (see *mt(1)*).

**Example 1:**

Example of archiving your entire Mach<sup>Ten</sup> 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 different save sets on a single tape using a blocking factor of 64, hardware compression, and 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
```

### 6.9.7.3 dump and restore

Complete and incremental backups of fast file systems may be done with *dump*. Recovery of files is done using *restore*. These tools are not useful on UNIX file systems. Details for using *dump* and *restore* are provided in the online man pages.

## 7.0 The Mach<sup>Ten</sup> Network File System (NFS)

The Mach<sup>Ten</sup> 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<sup>Ten</sup> 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<sup>Ten</sup> extends NFS to include the entire Macintosh family.

Mach<sup>Ten</sup> systems may be used as NFS clients or NFS servers. As a client, a system running Mach<sup>Ten</sup> can use files stored remotely on NFS networked machines as though they were local files. As a server, a Mach<sup>Ten</sup> system can store files for other NFS clients. If two Mach<sup>Ten</sup> servers exist in a single environment, they can be set up to be each other's client.

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

## 7.2 How NFS Works

The Mach<sup>Ten</sup> 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<sup>Ten</sup> 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 (RPCs) 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)* 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<sup>Ten</sup> 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<sup>Ten</sup> 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* command, 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:

```
/sbin/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 the server
server_dir	the directory that the client wishes to mount
client_mount-point	the directory that the client uses to access <i>server_dir</i>
type	nfs (for remote NFS mount)
options	mounting options
freq	0 (needs to be included, but not used in Mach <sup>Ten</sup> )
pass	0 (needs to be included, but not relevant to Mach <sup>Ten</sup> )

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<sup>Ten</sup> 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 startup (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 have the system set up for automatic mounting by using */etc/fstab*.

Configure Mach<sup>Ten</sup> as an NFS server and client by following these steps:

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<sup>Ten</sup> 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 as an NFS server, skip this step. If your system is to be used as an NFS server, restart Mach<sup>Ten</sup> 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 (i.e., 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<sup>Ten</sup> extends the traditional NFS service (accessible from UNIX applications) to include access to NFS volumes from Macintosh applications. This feature is called Desktop NFS (DTNFS) 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<sup>Ten</sup> 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. For example, 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<sup>Ten</sup> 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<sup>Ten</sup> 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<sup>Ten</sup> 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<sup>Ten</sup>'s NFS server provides *bwnfsd*, but not *pcnfsd*.

## 7.4.1 Desktop NFS Volume Mounting

Mach<sup>Ten</sup> DTNFS is, in essence, an NFS client with special capabilities. The previous section describing NFS server and client setup applies to DTNFS as well. *dt nfs* 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<sup>Ten</sup> system at mount point */mnt*. The command:

```
dtmount /mnt
```

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.

Drag the DTNFS volume to the Trash to remove it from the desktop. 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 *dtfst*:

```
omar:/space /mnt nfs dtfst,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<sup>Ten</sup> NFS servers, the locking daemon is *bwnfsd*, and the option flag supplied to *mount* (or placed in */etc/fstab*) to enable locking is *bw\_lock*:

```
mount -t nfs -o dtfst,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 dtfst,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<sup>Ten</sup>, the command:

```
/usr/sbin/bwnfsd -A
```

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

### 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* 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
```

### 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 that use this Desktop NFS 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* or *lockd*.

The mechanisms to mount Desktop NFS 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. */etc/rc* will then 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 Desktop NFS 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 Desktop NFS 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 Desktop NFS volume added to the desktop. This username and password are authenticated by the server. 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* 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<sup>Ten</sup> system for a particular user, and it was desirable to mount the Desktop NFS 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<sup>Ten</sup> system for a particular user, and it was desirable to mount the Desktop NFS 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<sup>Ten</sup> 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<sup>Ten</sup> *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 behavior 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” or “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 Desktop NFS

With 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 Desktop NFS it is not possible for a non-*root* user to change the owner of a file. Attempting to do so will result 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 file's name.

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

A Mach<sup>Ten</sup> 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. Instead, 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<sup>Ten</sup> application. This resource may be edited using *ResEdit*. *Dext* 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 Desktop NFS 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 Configuring NIS Under Mach<sup>Ten</sup>

NIS (Network Information Service), formerly called Yellow Pages or YP, is a client and server system developed by Sun Microsystems that supports remote data access to various system parameters. NIS clients are able to make network-based requests to a centralized set of NIS servers for user and password, host name, internet names and addresses, and other UNIX system-specific information.

Mach<sup>Ten</sup> has a pre-installed NIS client implementation that, with a relatively simple configuration, will support Mach<sup>Ten</sup> NIS client operations. This section describes the Mach<sup>Ten</sup> NIS client configuration process.

In the context of Mach<sup>Ten</sup>, NIS and NIST refer to version 3 of the NIS protocol. YP is used to refer to an earlier version 2 of the protocol. Mach<sup>Ten</sup> has implementations of both version 2 and version 3 of the NIS protocol. Configuration of NIS and NIST parameters provides for setup and operation of an NIS client with a version 3 NIS server. Configuration of YP parameters provides for version 2 NIS server operations. You may specify either or both configurations, according to your needs.

Data to resolve NIS client requests may reside on one or several local or remote databases. By default, the local database is searched first. If a match to the request is not found, one or more remote databases are searched. The Mach<sup>Ten</sup> NIS client configuration provides for specifying the order in which databases are searched.

### 8.1 NIS Basic Configuration Steps

To enable NIS operations, a system must be configured with the information to contact one or more NIS servers. The order and types of search for an NIS server can be specified as part of the Mach<sup>Ten</sup> NIS client configuration. The NIS server specifications for NIS version 2 YP operations are contained in the */etc/yp.conf* file. The NIS server specifications for NIS version 3 NIS and NIST operations are contained in the */etc/nis.conf* file. The order in which databases are searched is defined in the */etc/nsswitch.conf* file. After a set of NIS servers and their search order are specified, the system configuration file */etc/rc.conf* must be modified to start two NIS client software daemons — *portmap* and *ypbind*. After those configuration steps are complete, NIS client operations should be available.

## 8.2 Set YP Configuration in /etc/yp.conf

To configure operations for a version 2 NIS server, the */etc/yp.conf* file is modified. The */etc/yp.conf* file contains *ypserver* and *domainname* entries. To define a YP NIS server, specify the internet address or name of the server on a *ypserver* line in the */etc/yp.conf* file. There may be multiple *ypserver* entries in an */etc/yp.conf* file. The servers are contacted in the order in which they are entered in the file until a match for the requested data is found. *ypserver* entries have the following form:

```
ypserver      10.0.0.28
ypserver      10.0.0.29
```

Setting a *domainname* entry in */etc/yp.conf* maps all operations of a specific domain to the specified *ypserver*:

```
domainname    tenon.com
```

## 8.3 Set NIS Configuration in /etc/nis.conf

The */etc/nis.conf* file controls the sequence and internet addresses of a list of NIS version 3 servers. Like the */etc/yp.conf* file, the */etc/nis.conf* file contains *nissserver* entries and *domainname* entries. The *nissserver* entries define NIS version 3 servers to contact, and the order in which they are contacted. The *domainname* entries override server specifications and map all operations for a specified *domainname* to a specified *nissserver*. The *nissserver* entries have the following form:

```
nissserver    10.0.0.30
nissserver    10.0.0.31
```

The *domainname* entries have the following form:

```
domainname    tenon.com
```

## 8.4 Set NIS Database Search Order

The order in which data files and remote data sources are searched to satisfy a request is important. By providing for an ordered search of data, local data can specify overrides to host names, user names or other information that is special to local operations. If a request is not satisfied by looking locally, remote data sources are searched in the order specified in their configuration files.

The `/etc/nsswitch.conf` file contains line-by-line entries that specify the data sources and the search order for each of the types of NIS data requests that are supported by the Mach<sup>Ten</sup> NIS and YP software. In the `/etc/nsswitch.conf` file, `nisplus` and `nis+` are used to refer to NIS version 3 protocol operations. `nis` and `yp` are used to refer to NIS version 2 protocol operations.

### 8.4.1 The Default `/etc/nsswitch.conf` File

The following is a copy of the default `/etc/nsswitch.conf` file:

```
#
# /etc/nsswitch.conf
#
# An example Name Service Switch config file. This file should be
# sorted with the most-used services at the beginning.
#
# The entry '[NOTFOUND=return]' means that the search for an
# entry should stop if the search in the previous entry turned
# up nothing. Note that if the search failed due to some other reason
# (like no NIS server responding) then the search continues with the
# next entry.
#
# Legal entries are:
#
#      nisplus or nis+      Use NIS+ (NIS version 3)
#      nis or yp           Use NIS (NIS version 2), also called YP
#      dns                 Use DNS (Domain Name Service)
#      dbm                 Use DBM access
#      files               Use the local files
#      [NOTFOUND=return]   Stop searching if not found so far
#
passwd:      dbm files nisplus nis
shadow:     files nisplus nis
group:      files [NOTFOUND=return] nisplus nis
```

```
hosts:      files dns [NOTFOUND=return] nisplus nis

services:  files [NOTFOUND=return] nisplus
networks:  files [NOTFOUND=return] nisplus
protocols: files [NOTFOUND=return] nisplus
rpc:       files [NOTFOUND=return] nisplus
ethers:    nisplus [NOTFOUND=return] files
netmasks: nisplus [NOTFOUND=return] files
bootparams: nisplus [NOTFOUND=return] files

netgroup:  nisplus

publickey: nisplus

automount: files nisplus
aliases:   files nisplus
```

Lines in the `/etc/nsswitch.conf` file that begin with a “#” character designate comments which are ignored by the software that processes the file. Lines that begin with a “name” followed by a “:” are lines that specify the order in which different databases are searched to satisfy a request for data of the type specified by the “name” parameter. The `passwd` “name” specifies the database search order for user and password requests. The `hosts` “name” specifies the database search order for host names and addresses. The `files` search order designation refers to a local data file. The `dbm` designation refers to a local database file. The `nisplus` and `plus` designations refer to remote NIS requests to NIS version 3 and NIS version 2 protocol servers. The `NOTFOUND=return` file search order designation specifies that a search should terminate if all previous database entities have been checked and have responded negatively to a request. If a search failed for some other reason, such as NIS server not responding or unavailable, the search continues with the next entry.

The previous example contains the default settings included with Mach<sup>Ten</sup> NIS. The order of these settings is fairly standardized and typically requires little or no modification.

## 8.5 Testing the Basic NIS Configuration

After */etc/nis.conf*, */etc/yp.conf* and */etc/nsswitch.conf* have been reviewed and their configurations set appropriately, it is possible to test the basic NIS network configuration for proper operation. First the *portmap* and *ypbind* software daemons must be started to facilitate the testing. The commands:

```
root@ppc0# /usr/sbin/portmap
root@ppc0# /usr/sbin/ypbind
root@ppc0# ps -ax
  PID TT  STAT   TIME       COMMAND
    0  ?    R           (kernel idle)
    1  ?    S           /sbin/init
   63  ?    S    0:00       /usr/sbin/syslogd
   80  ?    S    0:00       /usr/sbin/inted
  232  ?    S    0:00       /usr/sbin/portmap
  235  ?    S    0:00       ypbind (slave)
  236  ?    S    0:00       ypbind (slave)
   86 co   S    0:00       /usr/bin/wind
   88 co   S    0:03       -csh (tcsh)
  237 w0   R    0:00       ps -ax
```

start the */usr/sbin/portmap* and */usr/sbin/ypbind* software daemons. The *ps -ax* command outputs a list of the executing programs and, in this example, shows that the *portmap* and two *ypbind* (slave) instances are ready for NIS operations.

To test that the daemons can properly contact a YP or NIS server, use the *ypmatch* or *ypcat* applications. The commands:

```
root@ppc0# ypmatch newton passwd
newton: .w.Xh2GK08Aw:507:507:Jon Newton:/home/newton:/bin/csh
root@ppc0# ypcat hosts
10.1.1.2      ppc100
127.0.0.1    host.foo.com
127.0.0.1    localhost
10.1.1.1     lamp100
```

are used to exercise the NIS component of the Mach<sup>Ten</sup> configuration. The *ypmatch newton passwd* contacts the first NIS server specified in the */etc/nis.conf* file and requests a match for user *newton* in the password database. The printed information is the corresponding *newton* entry from the centralized database.

The *ypcat hosts* command causes a dump of the complete database to be printed on the local screen. In the example above, the *hosts* database was printed.

Both *yycat* and *yymatch* have manual page entries with more information about running the applications and other parameters. The manual pages for *yycat* and *yymatch* are available by typing the *man yycat* or *man yymatch* commands.

## 8.6 Configure Mach<sup>Ten</sup> for Automatic Portmap and Ypbind Startup

After proper operation has been confirmed, the Mach<sup>Ten</sup> system should be configured to automatically start *portmap* and *ypbind* when Mach<sup>Ten</sup> is launched. This is done by modifying the */etc/rc.conf* file and setting the *START\_portmap* and *START\_ypbind* configuration settings to the string "yes".

```
START_portmap="yes"  
START_ypbind="yes"
```

Any subsequent restarts of the configured Mach<sup>Ten</sup> system will automatically cause *ypbind* and *portmap* to be started and facilitate the automatic access to NIS data.

## 9.0 Printing

Mach<sup>Ten</sup> 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<sup>Ten</sup> your Macintosh can take full advantage of UNIX-style printer sharing and printer administration.

With this hybrid system, interconnected Mach<sup>Ten</sup> and other UNIX systems are able to spool text and PostScript to traditional UNIX printers, as well as AppleTalk LaserWriter and ImageWriter printers. Mach<sup>Ten</sup> 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 LaserWriter server daemon (*lwsrv*), Mach<sup>Ten</sup> can emulate a named AppleTalk LaserWriter. This allows remote Macintosh applications to direct printer output to a Mach<sup>Ten</sup> system which will, in turn, spool the output as a Mach<sup>Ten</sup> print job.

### 9.1 Mach<sup>Ten</sup> 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<sup>Ten</sup>, every Macintosh is able to be more efficient in its printing by becoming a spooling client/server system.

In a Mach<sup>Ten</sup> 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<sup>Ten</sup> is a multi-tasking system, the server continues to run in the background and feeds the spooled data to a designated printer one file at a time. In a networked printing environment, the server forwards the spooled data to a remote server for printing from the remote system. Thus, Mach<sup>Ten</sup> provides not only local printer spooling, but remote printer spooling as well.

The client program in the Mach<sup>Ten</sup> print spooling system is *lpr(1)*. The background server program, also known as the print daemon, is *lpd(8)*. Detailed information on *lpr*, *lpd* and other programs used by the Mach<sup>Ten</sup> 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<sup>Ten</sup> CD-ROM.

### 9.1.1 The Print Spooler Database

The */etc/printcap* file (see *printcap(5)*) is the master database for the Mach<sup>Ten</sup> printing system. It describes all printers directly attached to your system, as well as printers accessible across the network. Mach<sup>Ten</sup> is shipped with an initial */etc/printcap* file (that will print to a Chooser-selected LaserWriter over AppleTalk) which you may need to modify, depending on your printing configuration.

## 9.2 Local Printing

In order to print with a minimum of system configuration, Mach<sup>Ten</sup> comes configured for local spooling of print data. With local spooling, a Mach<sup>Ten</sup> system acts as both a print spooling client and server.

Mach<sup>Ten</sup> 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.



## 9.2.1 Printing Text Files to a PostScript LaserWriter on AppleTalk

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

```
lp <file_name>
```

*lp* is a Mach<sup>Ten</sup> shell script which invokes *pstext(1)*, generating formatted PostScript 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 the Chooser. This is the default configuration for Mach<sup>Ten</sup>. 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.

## 9.2.2 Printing to an ImageWriter or a DeskWriter

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

1. To print to a locally connected ImageWriter or DeskWriter, perform the following steps:

For an ImageWriter:

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:
```

For a DeskWriter:

Edit the `/etc/printcap` file and uncomment the entry starting with `"dw|lp|DeskWriter:\\"`. 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. If the printer is connected to the printer port, change the line that says `"lp=/dev/ttyfa"` to read `"lp=/dev/ttyfb"`. If the printer 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 order to start *lpd*, you must “su” to *root* and start the daemon by giving the command */usr/sbin/lpd*. If you print from Mach<sup>Ten</sup> frequently, edit the */etc/rc.conf* file to start *lpd* automatically.



The ImageWriter is not selected with the Chooser, but rather 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. They 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.

## 9.3 Remote Printing

In a networked printing environment, Mach<sup>Ten</sup> supports spooling to a remote UNIX or Mach<sup>Ten</sup> system with an attached printer via *lpd*. The local *lpd* uses the TCP protocol to communicate with the remote *lpd*. Your Mach<sup>Ten</sup> 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 the */etc/printcap* file.

### 9.3.1 Spooled Printing to a Remote UNIX Printer

To set up Mach<sup>Ten</sup> 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<sup>Ten</sup> 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 (“#”)) 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 the first line and the remote host machine name on the second line (*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 in 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 the *chmod(1)* command.

5. Start the line printer daemon:

```
lpd
```

6. Print:

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



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

### 9.3.2 Receiving Remote Print Jobs

Mach<sup>Ten</sup> 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<sup>Ten</sup> 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 “9.2 Local Printing”.
2. To enable remote print access, add any remote host names that will use the printer as a remote printer to 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>
```

## 9.4 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)*) and section “5.4 Tailoring Your UNIX Environment” for more information). 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 this:

```
: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<sup>Ten</sup> system will retrieve the name of the printer from the information saved by the Chooser.

## 9.5 Extending Remote Printing to Macintosh Applications

The Mach<sup>Ten</sup> UNIX spooling capability is not only available for UNIX applications, but is extended to all Macintosh applications (even Macintosh applications on non-Mach<sup>Ten</sup> systems) through a LaserWriter server named *lwsrv*(8). *lwsrv* is a background Mach<sup>Ten</sup> program that identifies itself to the local AppleTalk zone as a LaserWriter. Remote Macintoshes can select this “synthetic LaserWriter” with their Choosers as an output device. 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.



**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<sup>Ten</sup> 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<sup>Ten</sup> Macintosh and Macintoshes not running Mach<sup>Ten</sup>. In this example, the Printer and the Sun workstation are on Ethernet, the Mach<sup>Ten</sup> 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 Printer via Max and *lwsrv*.

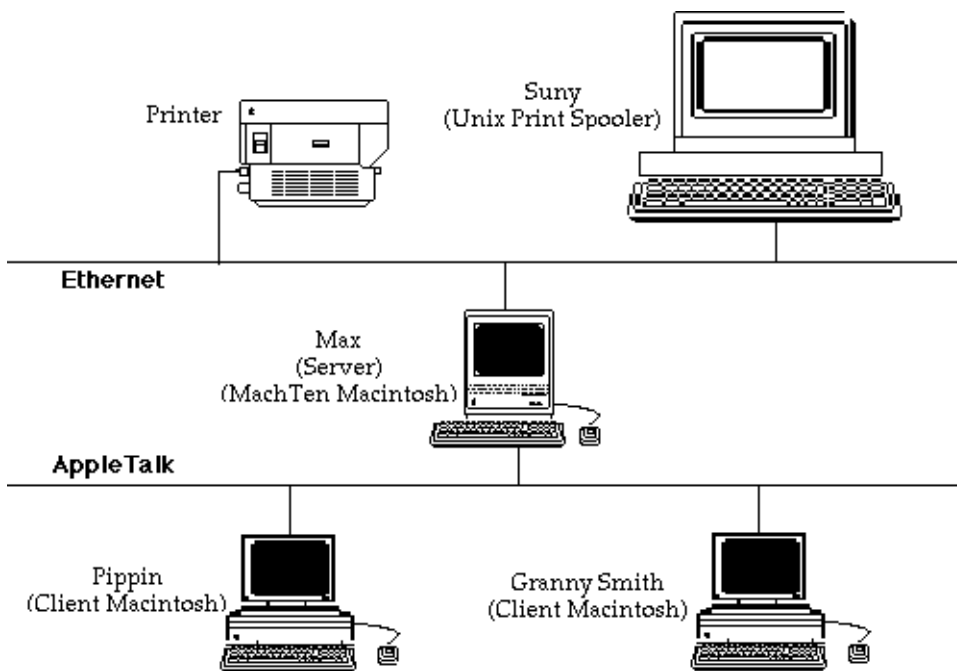


Figure 26. Spooling Printer Output to UNIX Printers Using *lwsrv*

### 9.5.1 *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 a QuickDraw-to-PostScript macro that this Macintosh user has on the client Macintoshes (Granny Smith and Pippin in the above example). Tenon supplies font files that will work for LaserWriter or LaserWriter Plus printers, as well as 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, MacOS System 7 or MacOS System 8.



### 9.5.1.1 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 `"*"`.

### 9.5.1.2 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.

## 9.5.2 Enabling *lwsrv*

To set up the Mach<sup>Ten</sup> spooling system to act as a LaserWriter emulator for remote Macintosh applications, perform the following steps:

1. Configure your printer following the instructions in sections “9.2 Local Printing” or “9.3 Remote Printing”.



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<sup>Ten</sup> system is using a remote *lwsrv* to support local *lpd* operations, the local */etc/printcap* must have an “*at*” entry and an “*rl*” entry in the form:

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

indicating the remote LaserWriter is a “synthetic system” and is unable to support normal maintenance PostScript requests such as PostScript page count. Refer to section “9.4 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 (see “Figure 26. Spooling Printer Output to UNIX Printers Using *lwsrv*”), 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 “9.5.1.1 Font Lists”.

3. Start *lwsrv*. The command:

```
/usr/sbin/lwsrv\  
-n mylaser \  
-a /usr/lib/adicts \  
-f /usr/lib/LWPlusFonts \  
-l /var/adm/lwsrv-errors -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-errors* and local *lpd* printer *lp*. (If you want *lwsrv* to be invoked every time you start Mach<sup>Ten</sup>, 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.

## 9.6 Print Job Status Mail Notification

If there is an access violation or print filter error, any Mach<sup>Ten</sup> *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.

## 9.7 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.

```
lpq
lp is ready and printing
Rank      Owner      Job  Files      Total Size
active    johnson     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.

## 10.0 Networking with Mach<sup>Ten</sup>

Mach<sup>Ten</sup> is rich in features that promote the sharing of data and processor resources with remote systems. Mach<sup>Ten</sup> communications software supports the exchange of data, remote timesharing of resources, and global electronic mail with a wide variety of systems. “Figure 27. Resource Sharing: FTP, mail, rlogin” shows a Mach<sup>Ten</sup> system with three Mach<sup>Ten</sup> windows open. The user has an *rlogin* session in one window, an *ftp* (File Transfer Protocol) in another, and has just sent a message in a third window. Since Mach<sup>Ten</sup> 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).

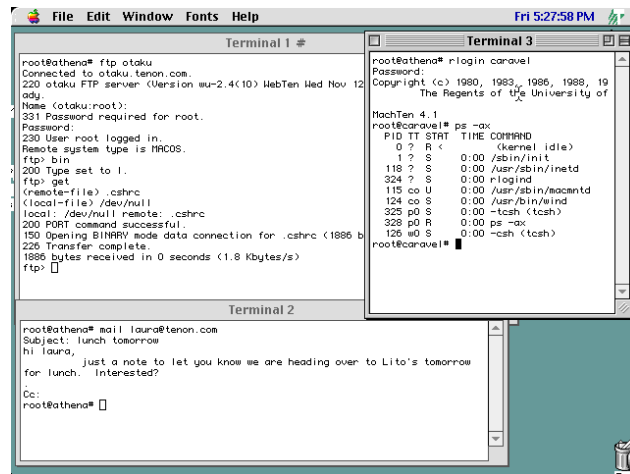


Figure 27. Resource Sharing: FTP, mail, rlogin

## 10.1 Using OpenTransport or Replacing OpenTransport

Mach<sup>Ten</sup> has been programmed with two alternative TCP/IP protocol implementations. The first is an implementation of the standard Mach<sup>Ten</sup> sockets interface which has been mated with MacOS OpenTransport. In this configuration, calls to the Mach<sup>Ten</sup> sockets interface are translated into one or more calls onto the MacOS OpenTransport TCP/IP implementation. In addition, the results of any OpenTransport calls are translated into Mach<sup>Ten</sup> sockets operations. This is the default Mach<sup>Ten</sup> configuration. It provides for complete compatibility between Mach<sup>Ten</sup> networking applications and other MacOS networking applications.

The second TCP/IP configuration provides for the replacement of OpenTransport with Tenon's Mach<sup>Ten</sup> TCP/IP stack. This implementation is an industrial-strength BSD UNIX TCP/IP implementation containing many features not found in OpenTransport, such as multi-link, multihoming.<sup>†</sup>

A third alternative is to simultaneously use both OpenTransport and Tenon's TCP/IP stack. The advantages and details for setting up this "dual" stack configuration are discussed in section "10.7 Dual Stack Configuration".

The TCP stack in Mach<sup>Ten</sup> is more robust, better-performing, and more fully featured than Apple's OpenTransport. When Mach<sup>Ten</sup> is running, Macintosh applications that are accustomed to using MacTCP will automatically use the TCP protocols in Mach<sup>Ten</sup>. "Table 4: Mach<sup>Ten</sup> TCP/IP Protocol Stack" shows the differences between Tenon's TCP stack and Apple's OpenTransport.

---

<sup>†</sup> OpenTransport 1.3 includes single-link, multihoming.

**Table 4: Mach<sup>Ten</sup> TCP/IP Protocol Stack  
(Comparison with OpenTransport)**

	OpenTransport	Mach <sup>Ten</sup> 's TCP
<b>Performance</b>	Caters to LANs.	Buffering strategies, window management, and packet re-transmission algorithms are designed to maximize both LAN and WAN performance.
<b>Multihoming</b>	OpenTransport 1.3.  Supports single-link, multihoming.	Supports multi-link, multihoming. Support for multiple network interfaces allows you to have redundant network connections and allows Mach <sup>Ten</sup> to act as a gateway to forward IP packets.
<b>Maximum Number of TCP Connections</b>	No limitation.	No limitation.
<b>Control Over Interface Options</b>	No intrinsic control. Many individual shareware packages are available to fulfill various control functions.	Provides a range of management options, such as ARP table management, statistics reporting and reconfiguration without rebooting.
<b>Routing Control</b>	Supports single default route.	Allows many routing entries and supports dynamic routing configuration via <i>routed</i> and <i>gated</i> .
<b>Domain Name Resolution</b>	Supports single (remote) domain name server and uses a non-standard "hosts" file	Supports multiple domain name servers (local and remote) and uses a standard "hosts" file.

	OpenTransport	Mach <sup>Ten</sup> 's TCP
<b>Simultaneous Support of UNIX and Macintosh Applications</b>	Yes, when used with Mach <sup>Ten</sup> .	Yes, unless the application is OpenTransport-only.
<b>Multicast</b>	No.	Yes.
<b>Loopback Operation</b>	Yes.	Yes.
<b>IP Aliasing</b>	Yes.	Supports multiple IP addresses on a single network interface.

Mach<sup>Ten</sup> internetworking is designed to be self-configurable upon installation. However, the installation process assumes that you are installing Mach<sup>Ten</sup> 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 that need to be performed to set up or change your network environment and to keep your network running — “10.3 Using Mach<sup>Ten</sup> as an Internet Host”, “10.4 Using Mach<sup>Ten</sup> as a Domain Name Server”, and “10.6 Using Mach<sup>Ten</sup> as a Router”.



## 10.2 Configuring Tenon TCP or Dual Stack Networking

The choice of using OpenTransport, Tenon TCP or dual stack networking is controlled by the placement of three libraries in the *Tenon Kernel Libraries* folder. Table 5 shows the valid combinations for the location of these libraries. To change configuration, quit Mach<sup>Ten</sup>, use the Finder to re-position the libraries, and re-launch Mach<sup>Ten</sup>.

**Table 5: Networking Configuration Choices**

<b>Configuration</b>	<b>Tenon Kernel Libraries</b>	<b>Disabled</b>
Use OpenTransport (default)	OT SocketLib	SocketLib MacTCPdLib
Replace OpenTransport	SocketLib MacTCPdLib	OTSocketLib
Dual stack	SocketLib	OTSocketLib MacTCPdLib

## 10.3 Using Mach<sup>Ten</sup> as an Internet Host

Mach<sup>Ten</sup> 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<sup>Ten</sup> supports the full range of internet class A through C internet addresses, as well as internet subnetting capabilities. You must contact Network Solutions in Herndon, Virginia (<http://www.internic.net>) to obtain an internet address for your network. 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.

In Mach<sup>Ten</sup>, your host name and internet address are from information supplied in the Mach<sup>Ten</sup> 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<sup>Ten</sup> 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 “10.4 Using Mach<sup>Ten</sup> 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`.

### 10.3.1 The Mach<sup>Ten</sup> Networking Control Panel

Mach<sup>Ten</sup> allows for a multi-homed internet networking environment — permitting multiple independent networks to be connected to a single system. Mach<sup>Ten</sup> supports several physical interface types, including Ethernet, SLIP, PPP, TokenRing and LocalTalk, in addition to a logical local loopback interface. The Mach<sup>Ten</sup> 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<sup>Ten</sup> 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 “10.15.8 Using PPP” and “10.15.9 Using SLIP”.

### 10.3.2 Configuring Multiple Interfaces

Mach<sup>Ten</sup> 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<sup>Ten</sup> permits up to four Ethernet interfaces, named *ie0-ie3*. 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.

## 10.4 Using Mach<sup>Ten</sup> as a Domain Name Server

Mach<sup>Ten</sup> is designed to use the Internet Name Server protocol to translate host names into internet addresses. The Mach<sup>Ten</sup> 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<sup>Ten</sup>, the Mach<sup>Ten</sup> installer, in conjunction with the Mach<sup>Ten</sup> control panel, automatically sets up a table of host names and internet addresses in the file */etc/hosts*. If all of the appropriate information was available when you installed Mach<sup>Ten</sup> 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<sup>Ten</sup> CD) tells you how to set up a sub-domain 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.

### 10.4.1 Configuring Mach<sup>Ten</sup> as a Domain Name Server

If you want to configure one of your Mach<sup>Ten</sup> systems 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<sup>Ten</sup> 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 28. Example */etc/hosts* File”).

It is convenient to keep your active files in a subdirectory of */etc* (*/etc/nsdata*, for example). 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 startup. 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*.

```
# 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      mumm.some.com   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 28. Example */etc/hosts* File

The example boot file in `/etc/named/boot` follows:

```

; Boot file for YourZone primary master name server
;
; domain          YourZone
directory        /etc/named
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`.

## 10.4.2 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<sup>Ten</sup>, 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<sup>Ten</sup> control panel. The name server may be a `named` running on your local machine or on another network host. Mach<sup>Ten</sup> will automatically create the `/etc/resolv.conf` file when launched. If you want the resolver to also query a backup name server, simply edit the `/etc/resolv.conf` file and add the name server address in the form shown above.

## 10.5 Using Mach<sup>Ten</sup> as a Web Server

The Apache hypertext transfer protocol daemon (*httpd*) serves World Wide Web (WWW) documents from your Mach<sup>Ten</sup> 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<sup>Ten</sup> boot up, edit the */etc/rc.conf* file by changing the line `START_httpd` to `START_httpd="yes"`. Then restart Mach<sup>Ten</sup>.

For detailed documentation on Apache *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<sup>TM</sup>) at URL *http://localhost*, and for editing in the local file */etc/httpd/htdocs/MyHomePage.html*.

The BBEditLite 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 BBEditLite documents located in the *Utilities:Text Utilities* folder on the Mach<sup>Ten</sup> CD-ROM for more information.

### 10.5.1 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>
```

## 10.5.2 Multihoming Your Mach<sup>Ten</sup> Web Server

The Mach<sup>Ten</sup> *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<sup>Ten</sup> achieves this unique level of service in two ways:

First, Mach<sup>Ten</sup> 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.



## 10.6 Using Mach<sup>Ten</sup> as a Router

The Mach<sup>Ten</sup> internet routing service is typically used to connect Mach<sup>Ten</sup> systems on an AppleTalk network to Mach<sup>Ten</sup> systems and other UNIX systems on an Ethernet. By supplying an Ethernet interface to any Macintosh running Mach<sup>Ten</sup>, that Macintosh can serve as a router between a set of Ethernets and a set of AppleTalk networks. The Mach<sup>Ten</sup> system providing the routing service must be connected to both the AppleTalk network and the Ethernet. The other hosts (Mach<sup>Ten</sup>, UNIX and others) need to know which Mach<sup>Ten</sup> 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<sup>Ten</sup> the address of the routing server is the *route(8)* command. Mach<sup>Ten</sup> 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 “9.5 Extending Remote Printing to Macintosh Applications” of an AppleTalk network and an Ethernet each supporting Mach<sup>Ten</sup> hosts, as well as other hosts. All Mach<sup>Ten</sup> 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<sup>Ten</sup> 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<sup>Ten</sup> 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<sup>Ten</sup> 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<sup>Ten</sup> kernel to route all of the packets destined for the Ethernet to AppleTalk address “A.1”.

Non-routing Mach<sup>Ten</sup> systems refer to routers as “gateways”. Configure your system with a default gateway in the Networking screen of the Mach<sup>Ten</sup> control panel. Other routes may be added using the *route(8)* command.

## 10.6.1 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<sup>Ten</sup> control panel. If the setting of the check box is changed, the new setting takes effect after the system is rebooted.

## 10.7 Dual Stack Configuration

It may be convenient to configure both OpenTransport and the Mach<sup>Ten</sup> TCP/IP for simultaneous operations on the same system. This configuration requires configuring each TCP/IP with a different IP address. Using this configuration, a single Macintosh has two Internet Protocol (IP) addresses. The IP address configured for OpenTransport is contained in the OpenTransport TCP/IP control panel. The IP address configured for Mach<sup>Ten</sup> is contained in the Mach<sup>Ten</sup> control panel under the Networking sub-panel. Simply configuring different addresses in each of these control panels, and configuring Mach<sup>Ten</sup> for Mach<sup>Ten</sup> BSD TCP/IP operations, will provide for a configuration that allows Mach<sup>Ten</sup> networking applications to take full advantage of the strength of the Mach<sup>Ten</sup> TCP/IP stack. At the same time, the OpenTransport TCP/IP is fully configured and operational for any Macintosh applications that wish to take advantage of its capabilities.

There are a few things you should consider if you decide to use the dual stack configuration. First, more RAM will be required to run both stacks. Second, communication between the two stacks may be configured.

### 10.7.1 More RAM

More RAM will be used when running both TCP/IP implementations. You should be prepared to dedicate as much as 850K of extra RAM to simultaneous TCP/IP operations. This RAM will not be available to your applications.

### 10.7.2 Cross Talk

Since both TCP/IP implementations are running on the same physical interface on the same machine, under normal circumstances they cannot talk directly to each other. If you have no requirement for the OpenTransport applications to talk to Mach<sup>Ten</sup> applications (on the same machine), this may not be an issue. However, if such communications are necessary, there is a solution.

### 10.7.2.1 Routing Between Mach<sup>Ten</sup> and OpenTransport

Mach<sup>Ten</sup> allows the deft administrator to use a local router as an intermediary for packet exchange between the two TCP/IP stacks.

The trick is to configure OpenTransport with an IP address and a subnet mask where, from OpenTransport's point of view, the Mach<sup>Ten</sup> IP address will be on a different subnet than the OpenTransport IP address. This will force OpenTransport to direct all packets destined for Mach<sup>Ten</sup>'s TCP/IP to the router address specified in your OpenTransport TCP/IP control panel. Note that you should assign your OpenTransport IP address and subnet mask such that the OpenTransport IP address is on the same subnet as your default router.

On the Mach<sup>Ten</sup> side, launch the Mach<sup>Ten</sup> control panel and display the "Networking Preferences" dialog. Configure an IP address that now resides on a logical subnet different than the subnet containing the OpenTransport IP address and based on the OpenTransport subnet mask as configured in the OpenTransport TCP/IP control panel. No adjustment to the Mach<sup>Ten</sup> TCP/IP subnet mask to match the OpenTransport subnet mask is necessary; use your original subnet mask for Mach<sup>Ten</sup>.

Next you must adjust the Mach<sup>Ten</sup> TCP/IP routing behavior, routing local packets sent to your OpenTransport stack's IP address through your local gateway. To achieve this, you must manually edit the */nfs/etc/rc.local* file and add an entry to the bottom of the file that looks like this:

```
/sbin/route add host <OT-IP-Address> $GATEWAY 1
```

where <OT-IP-Address> is the OpenTransport IP "dot" address as configured in your OpenTransport TCP/IP control panel. This tells Mach<sup>Ten</sup>'s TCP/IP stack to direct all packets destined for your OpenTransport IP stack to the router. When packets arrive at the router from either Mach<sup>Ten</sup> TCP/IP or OpenTransport TCP/IP, it will take care of routing them back to the appropriate stack on your machine.

### 10.7.2.2 Using a Macintosh DNS Server on Mach<sup>Ten</sup>

If you use a Macintosh DNS server that uses OpenTransport, it will not be directly accessible to Mach<sup>Ten</sup>. Aside from the more elaborate router solution presented in section “10.7.2.1 Routing Between Mach<sup>Ten</sup> and OpenTransport”, a simple solution is to configure Mach<sup>Ten</sup> to do DNS lookups through another DNS server (perhaps a local ISP's). This remote DNS server will talk to your local OpenTransport-based DNS and will automatically pass the results back to Mach<sup>Ten</sup>. Since the other DNS server is outside of the local machine, it solves the problem by being able to talk to both TCP/IP addresses.

If you run no Macintosh TCP applications on your Mach<sup>Ten</sup> machine other than a DNS server, you may alternatively enable BIND DNS under Mach<sup>Ten</sup> and eliminate the Macintosh DNS server.

The DNS server Mach<sup>Ten</sup> queries to resolve domain names can be configured in the Mach<sup>Ten</sup> Preferences. Enter the IP address of the alternate server in the “DNS IP Address” field, and Mach<sup>Ten</sup> will automatically start requesting DNS information from that IP address. If you enable BIND DNS under Mach<sup>Ten</sup>, duplicate the IP address of the Mach<sup>Ten</sup> TCP/IP in the “DNS IP Address” field.

## 10.8 Using Mach<sup>Ten</sup> 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<sup>Ten</sup> Networking control panel and restart Mach<sup>Ten</sup>.



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

## 10.9 Using Electronic Mail

Mach<sup>Ten</sup> 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<sup>Ten</sup> system are able to send electronic mail messages to one another. In addition, users logged into a Mach<sup>Ten</sup> 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<sup>Ten</sup> 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<sup>Ten</sup> installation process, *sendmail* is configured to support direct mail activities on the local Macintosh, as well as communicate with other mail servers if the Macintosh and Mach<sup>Ten</sup> have network access. 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<sup>Ten</sup> 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 8.1 6/6/93.  Type ? for help.
"/var/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: Meeting"
```

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<sup>Ten</sup> CD-ROM will help you use all of the features of the Berkeley UNIX *sendmail* included with Mach<sup>Ten</sup>. In addition, there are public domain mailers, such as Eudora<sup>†</sup> and Pine 4, that work very well with Mach<sup>Ten</sup>.

---

<sup>†</sup> <ftp://ftp.qualcomm.com/eudora>



## 10.10 Using FTP

Mach<sup>Ten</sup> 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<sup>Ten</sup>-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. You, in turn, can transfer and retrieve files from any connected system that supports FTP. See *ftp(1)*.

Mach<sup>Ten</sup> will interoperate with Macintosh FTP applications such as Fetch.<sup>†</sup> Binary transfers between Mach<sup>Ten</sup> and Macintosh FTP applications should be performed in AppleSingle mode, the Macintosh foreign file format used by Mach<sup>Ten</sup>. Refer to the section “6.7 Accessing Macintosh Files from UNIX Applications” for more information.

## 10.11 Using Mach<sup>Ten</sup> 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 database to exclude the “*ftp*” line, or uncheck the “Enable Incoming Connections” box in the Mach<sup>Ten</sup> Networking control panel and restart Mach<sup>Ten</sup>. If you leave the anonymous FTP service in place, documents for public consumption should be placed in the */home/ftp/pub* folder.

---

<sup>†</sup> <ftp://ftp.dartmouth.edu/pub/software/mac>

## 10.12 Using Telnet

Mach<sup>Ten</sup> supports both user and server Telnet, enabling each Mach<sup>Ten</sup> 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<sup>Ten</sup> 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)*.

## 10.13 Using a Distributed File System (NFS)

Mach<sup>Ten</sup> 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<sup>Ten</sup> 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".

## 10.14 Using Berkeley r-series Commands

Mach<sup>Ten</sup> 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.

## 10.15 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<sup>Ten</sup> and UNIX systems. Mach<sup>Ten</sup> supports traditional UNIX use of these serial ports and provides a comprehensive set of applications for communications via these ports.<sup>†</sup>

### 10.15.1 Cabling and Connectors

The serial ports require a Macintosh-specific connector — a mini 8-pin connector. Macintosh dealers carry a modem cable that 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.

### 10.15.2 Matching Serial Ports to UNIX Devices

Mach<sup>Ten</sup> 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.

---

<sup>†</sup> Mach<sup>Ten</sup> supports additional serial ports provided by add-on serial port multiplexer boards. Refer to the “Additional Serial Ports” technical note in the `/pub/tech_notes` directory on [ftp.tenon.com](http://ftp.tenon.com) for more information.

### 10.15.3 Baud Rates Supported

UNIX systems traditionally support baud rates from 50 to 9600 baud. Macintosh hardware does not support baud rates below 300 baud, but it does support baud rates higher than 9600 baud<sup>†</sup>. To take advantage of these higher baud rates, Mach<sup>Ten</sup> translates requests for the traditional UNIX baud rates to the more useful set of baud rates supported by the Macintosh.

“Table 6: Baud Rates Supported” lists the actual baud rate used by Mach<sup>Ten</sup> when the traditional baud rates are requested.

**Table 6: Baud Rates Supported**

Requested Baud Rate	Traditional UNIX Name <devtty.h>	Mach <sup>Ten</sup> 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
... continued on next page		

---

<sup>†</sup> If Serial DMA is version 2.0.1 or later. (Serial DMA is included on the Mach<sup>Ten</sup> CD-ROM.)

Requested Baud Rate	Traditional UNIX Name <devtty.h>	Mach <sup>Ten</sup> Baud Rate
1800	B1800	1800
2400	B2400	2400
4800	B4800	4800
9600	B9600	9600
19200	EXTA	19200
38400	EXTB	38400

### 10.15.4 General Purpose Interface (GPI) Serial Cable

Mach<sup>Ten</sup> supports the Macintosh General Purpose Interface (GPI) serial input. With properly constructed cabling (see “Figure 29. General Purpose Interface (GPI) Serial Cable”), Mach<sup>Ten</sup> can simultaneously support hardware flow control and carrier detect modem control transitions.

#### Modem Cable Pinout DIN-8 to DB-25

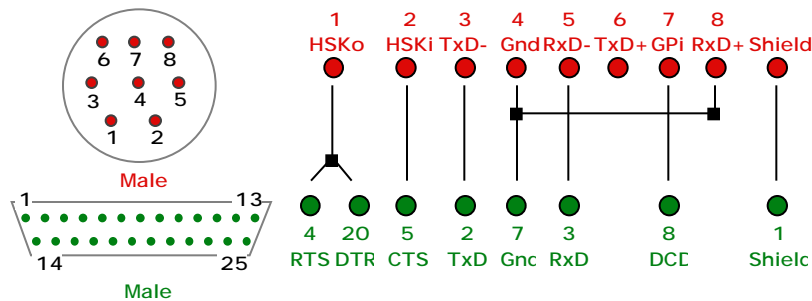


Figure 29. General Purpose Interface (GPI) Serial Cable

### 10.15.5 Connecting a Modem to a Mach<sup>Ten</sup> Serial Port

For incoming calls to a modem connected to the serial port, the governing Mach<sup>Ten</sup> 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 mode, 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<sup>Ten</sup> end of a connection to go onhook first at the end of a call, a Hayes compatible modem must be configured to heed the Data Terminal Ready (DTR) 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 by a Macintosh environment communications program. These programs normally take responsibility for configuring the modem. No special configuration action by the user is required beyond following the instructions for the particular program.

### 10.15.6 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<sup>Ten</sup>.

## 10.15.7 Configuring the System to Support an ASCII Terminal

After connecting an ASCII terminal with the proper cabling to the selected serial port, Mach<sup>Ten</sup> must be properly configured to communicate with the terminal. The */etc/ttys* and */etc/gettytab* files are the databases 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, type several carriage returns on the connected terminal. */etc/getty* is listening on that port and looking for a login string and password. Typing carriage returns will result in a login prompt appearing on the screen. Log out by typing “logout” or “^D”, depending upon what shell you are using.

## 10.15.8 Using PPP

The Point-to-Point Protocol (PPP) was developed to support serial communications in a multi-protocol 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, which include password authentication, PPP is the standard for serial communications by internet service providers.



### 10.15.8.1 Setting up Mach<sup>Ten</sup> PPP on Your Macintosh

This section describes how to configure your Mach<sup>Ten</sup> 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).

### 10.15.8.2 Configuring your Mach<sup>Ten</sup> 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<sup>Ten</sup> 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 dial-in 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.

1. Edit */etc/ttys* and set the status for "cm.in" to "on".
2. Signal the *init* process.

A Communications ToolBox dialog should appear.

3. Adjust the settings for your modem.

You should then get a "Waiting" message.

4. Dial in to your modem.

You should get a "Login:" prompt.

5. **IMPORTANT:** While you have the "Login:" prompt on the machine you are dialing in with, return to the Mach<sup>Ten</sup> pull-down the File menu and 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.

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

At this point you should have consistent dial-in access to Mach<sup>Ten</sup>.

The next step is to set up 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<sup>Ten</sup> users. The key is to use `/etc/pppserver` as the “shell” for PPP client accounts. The following command will set up 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<sup>Ten</sup> 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 your local subnet. The `proxyarp` option is used by `/etc/pppserver` to make this work.

To terminate a PPP connection from Mach<sup>Ten</sup>, 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.

### 10.15.8.3 Configuring your Mach<sup>Ten</sup> 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<sup>Ten</sup> 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 a Communications ToolBox dialog to appear.

1. Adjust the settings for your modem and dial out.

You should then see your remote system's prompt.

2. **IMPORTANT:** While you have your remote system's prompt, pull down the Mach<sup>Ten</sup> 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 */etc/remote* file controls which lines are accessible to the *tip* command. The first few lines already have entries. You will have to add more entries if you have more dial-outs. 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 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 dial-out 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 log in 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<sup>Ten</sup>, 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` that can be set to a static address for the client.

### 10.15.8.4 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 "0xffffffff00" 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.

## 10.15.9 Using SLIP

Mach<sup>Ten</sup> supports internet connections over full duplex serial lines using SLIP, the Serial Line Internet Protocol. Mach<sup>Ten</sup> can be configured to initiate connections to a remote SLIP server system. Mach<sup>Ten</sup> can also be configured to accept connections from hosts requesting SLIP service. Mach<sup>Ten</sup>'s SLIP capabilities, combined with Macintosh hardware and any of a number of commercially available modems, make an excellent platform for accessing large dial-up networks such as PSINet and UUNET. Other internet access providers offer SLIP services as well. Mach<sup>Ten</sup> 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<sup>Ten</sup>.

### 10.15.9.1 Configuring Mach<sup>Ten</sup> to Dial Out to SLIP Servers

Mach<sup>Ten</sup> 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 "10.15.10.5 Manual Connections to SLIP Servers".

During the SLIP session, the SLIP line, or "interface", becomes an integral part of the Mach<sup>Ten</sup> networking environment, running standalone or simultaneously with other network interfaces such as Ethernet or Appletalk configured into your Macintosh. Both Mach<sup>Ten</sup> 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.

### 10.15.9.2 Tip Activity Log

The file */var/log/aculog* is an administrative file that logs all dialing activity done by *tip*. It can be used to track who is calling which numbers and for how long.

### 10.15.9.3 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 modem's setup used to gain host access. Mach<sup>Ten</sup> 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:

<b>br# entry</b>	<b>Modem Baud Rate</b>
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 ( <i>/dev/ttya</i> ) is the default setting. You may also use the printer port ( <i>/dev/ttyb</i> ); however, the 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, <i>tip</i> will use the system name to look up a phone number in the <i>/etc/phones</i> database. (See <i>phones(5)</i> ).
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 <i>tip</i> 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 "10.15.9.5 Tip Login Script" for details on the script's command format.

#### 10.15.9.4 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 Computer 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<sup>Ten</sup> device name indicates hardware flow control on that serial port.

### 10.15.9.5 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/sliphome/login.script.joe` is provided as an example authentication dialogue with a remote UNIX *login(1)* shell. The file `/etc/sliphome/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:

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

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.

### 10.15.9.6 Header Compression

Mach<sup>Ten</sup> will enable SLIP header compression for initiating SLIP sessions if a “: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. The dial-in server configuration for Mach<sup>Ten</sup> systems is described below.

### 10.15.9.7 Setting a Default Route

Mach<sup>Ten</sup> will make the remote host in a SLIP connection the default route if no default route exists on your system and if a “: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<sup>Ten</sup> system are directed to the default route.

### 10.15.9.8 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 ensure good interactive performance while minimizing the header cost. Adjustments to the MTU can be made in the Networking screen of the Mach<sup>Ten</sup> control panel. The MTU setting is applied to all configured SLIP interfaces. The use of header compression (see section "10.15.9.6 Header Compression") is recommended for all MTU settings where supported by your SLIP server.

### 10.15.10 Configuring Mach<sup>Ten</sup> as a Dial-In SLIP Server

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

#### 10.15.10.1 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<sup>Ten</sup>) is the recommended port. The "printer" port (*/dev/ttyb*) is also acceptable. Note, however, that use of the printer port is usually reserved for AppleTalk.

To make Mach<sup>Ten</sup> 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<sup>Ten</sup> 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<sup>Ten</sup>. To enable flow control on the serial port, change the `ttya` field to `ttyfa`. The “f” in the Mach<sup>Ten</sup> device name indicates hardware flow control on that serial port. You must either send a “hang-up” signal (see `kill(8)`) to the `init` process (see `init(8)`) or restart your Mach<sup>Ten</sup> system to activate any modifications made to `/etc/ttys`.

## 10.15.10.2 Login Authentication

When a remote SLIP host initiates a connection on a configured serial port, Mach<sup>Ten</sup> will start the `login(1)` program, which prompts the remote host for a user name 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<sup>Ten</sup> `/etc/master.passwd` file contains a default account with no password for user “Sjoe”:

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

Modify this entry as necessary for your SLIP configuration.

### 10.15.10.3 The Sliplogin Program

*sliplogin(8)* in `/usr/sbin` 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/sliphome/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/sliphome/slip.hosts`. The default initialization script is `/etc/sliphome/slip.login`. However, if particular hosts need special initialization, the file `/etc/sliphome/slip.login.loginname` will be executed instead (if it exists).

The example `/etc/sliphome/slip.hosts` file from the distribution is shown below:

```
#      @(#)slip.hosts  8.1 (Berkeley) 6/6/93
#
# login    local-addr  remote-addr  mask          opt1
#                               (normal,compress,noicmp)
#
Schez     vangogh      chez         0xffffffff00  compress
Sjun      vangogh      128.32.130.36 0xffffffff00  normal
Sleconte  vangogh      leconte      0xffffffff00  compress
Sleeb     vangogh      leeb         0xffffffff00  compress
Smjk      vangogh      pissaro-sl   0xffffffff00  compress
Soxford   vangogh      oxford       0xffffffff00  compress
```

### 10.15.10.4 SLIP Session Termination

On session termination, *sliplogin* will automatically remove the SLIP interface from the list of Mach<sup>Ten</sup> network interfaces. Mach<sup>Ten</sup> will reset the serial port to the configuration specified in `/etc/ttys` in preparation for a new SLIP session.

### 10.15.10.5 Manual Connections to SLIP Servers

Some SLIP servers may not support the automated functions of *tip -s* (dial-up terminal servers or other non-BSD UNIX systems, for example). In this case, it may be necessary to perform some of the SLIP setup 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 setup steps below.

1. Configure an abbreviated entry in */etc/remote* as follows:

```
sliphost:\  
dv=/dev/ttyfa:br#2400 (Example baud rate (br#) setting)
```

2. Set up the *tip* activity log described in section “10.15.9.1 Configuring Mach<sup>Ten</sup> to Dial Out to SLIP Servers” if you want a log of your session.

3. Invoke *tip* without the *-s* option

```
tip sliphost
```

4. Dial out to the remote host (this example uses a modem with an “at” command set, dialing to a hypothetical phone number preceded by “1” and an area code):

```
atdt15185551212
```

*tip* may not immediately notice if the number is busy, but will time out waiting for a reply.

Subsequent steps in the manual SLIP connection process achieve the SLIP configuration that would otherwise be done automatically by *tip*.



## 10.15.10.6 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, and then give the command `slip`. With BSD systems, the system administrator usually creates a special login account 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.

## 10.15.11 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 login. 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 does not 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)`:

```
/sbin/slattach /dev/ttyfa <baud rate>
```

"<baud rate>" sets the speed of the connection (2400 baud, for example). 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:

```
/sbin/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:

```
/sbin/route add net default <gateway address> 1
```

### 10.15.11.1 Testing Connectivity

Once SLIP is running, the Mach<sup>Ten</sup> 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.

## 10.15.11.2 Stopping SLIP

When the SLIP session is over, it is desirable to undo the configuration done to bring SLIP up, especially if the Mach<sup>Ten</sup> 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.

1. Delete the routing table entry:

```
/sbin/route delete net default <gateway address>
```

2. Bring down the SLIP interface:

```
/sbin/ifconfig sl0 down
```

3. Kill the *slattach* process:

```
kill -9 `ps -ax | egrep slattach | egrep -v egrep | awk '{print $1}'`
```

4. Hang up the phone connection by powering off the modem.

### 10.15.11.3 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

/sbin/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.

/sbin/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.

/sbin/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.

/sbin/route delete net default ${1:r}.1

#Bring down the SLIP interface.

/sbin/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."
```



## 11.0 Mach<sup>Ten</sup> Programming Environment

Mach<sup>Ten</sup> provides a rich UNIX software development environment. The Mach<sup>Ten</sup> 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.

### 11.1 Mach<sup>Ten</sup> Development Tools

#### 11.1.1 Programs, Libraries and Include Files

Mach<sup>Ten</sup> consists of a combination of programs, libraries and *include* files from BSD4.4 (hereafter simply referred to as BSD), GNU, Tenon Intersystems, and other sources. Most programs reside in */usr/bin* and *include* files in */usr/include*. Programs, libraries and *include* files targeting a particular system architecture are stored in the sub-directories *bin*, *lib*, and *include* of a target directory in */usr*. For the Power Macintosh, the target directory is named */usr/macppc*. Support for other target architectures is planned for future releases. The tools in the target directories run on the host system, but produce objects and executables that run on the target system. Environment variables and symbolic links will help determine which tools are run.

Mach<sup>Ten</sup> 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	
<i>flex</i>	Generates lexical analysis programs
<i>indent</i>	Formats C code
<i>yacc</i>	“yet another compiler-compiler”

<b>Tools for Building Libraries and Programs</b>	
<code>ar</code>	Library archive utility
<code>as</code>	PPC assembler
<code>cpp</code>	GNU preprocessor
<code>g++</code>	Wrapper for calling <code>gcc</code> with appropriate options for compiling and loading C++ objects
<code>g77</code>	Wrapper for calling <code>gcc</code> with appropriate options for compiling and loading Fortran objects
<code>gcc</code>	GNU compiler; forks the appropriate tools for compiling and loading objects
<code>gdb</code>	Symbolic debugger
<code>gnatmake</code>	GNAT <i>make</i> program
<code>ld</code>	Combines several object files into one, resolving external references and searching libraries
<code>make</code>	Executes a scripted set of commands to update one or more target files; GNU version
<code>nm</code>	Prints lists of symbols in object files
<code>pmake</code>	Executes a scripted set of commands to update one or more target files; BSD version
<code>ranlib</code>	Converts archives to a form which <code>ld</code> can load more rapidly
<code>restool</code>	Provides access to Macintosh file resources
<code>setstackspace</code>	Adjusts size of program stack



<b>Compiler Engines</b>	
cc1	GNU C language compiler
cc1plus	GNU C++ language compiler
f771	GNU Fortran language compiler
gnat1	GNU Ada language compiler
cc1obj	GNU Objective-C compiler
<b>Libraries</b>	
libc	The C library
libobjc.a	The Objective-C library
libcap.a	Columbia AppleTalk Protocol library
libcompat.a	Contains obsolete BSD4.3 functions for compatibility
libcurses.a	Old screen functions with cursor motion
libf2c.a	The GNU Fortran library
libfl.a	The library for <i>flex</i>
libg++.a	The GNU C++ library; includes <i>libio.a</i> functions
libgdbm.a	GNU database management functions
libgnat.a	The GNU Ada library
libgthreads.a	POSIX Threads functions
libio.a	New GNU C++ <i>iostream</i> classes
libiostream.a	Old GNU C++ <i>iostream</i> classes; uses <i>libio.a</i> instead
libkaffe-agent.a	<i>kaffe</i> support library
libkaffe-native.a	<i>kaffe</i> support library
libkaffe-net.a	<i>kaffe</i> support library
libkaffe-vm.a	<i>kaffe</i> support library

<b>Libraries (continued)</b>	
libkvm.a	Kernel memory interface functions
libm.a	Math library
libncurses.a	New screen functions with cursor motion
libresolv.a	Domain name server interface functions
librpc.a	Remote procedure call library
librx.a	GNU regular expression functions
libstdc++.a	GNU ANSI C++ library
libterm.a	Terminal independent operation library package
libutil.a	Daemon support functions
liby.a	The library for <i>yacc</i>
<b>Tools for Building X Window Programs</b>	
imake	X platform independent <i>make</i> environment
libFS.a	Font Service library for X font clients
libICE.a	Inter Client Exchange library
liboldX.a	X11R4 support library
libolgx.a	OpenLook graphics library
libPEX5.a	PHIGS Extensions to X
libSM.a	X Session Management library
libX11.a	Core X protocol library
libXau.a	X security authorization library
libXaw.a	The Athena widget set of buttons, pull-down menus, labels, etc.
libXdmcp.a	X Display Manager Control Protocol library
libXext.a	X extensions library

Tools for Building X Window Programs (continued)	
libXi.a	Alternative input device extension library
libXIE.a	X Image Extensions library
libXmu.a	X miscellaneous utilities library
libXpm.a	X pixmap image library
libXt.a	X Intrinsics toolkit library
libXtst.a	X Test library
libXview.a	X View library

### 11.1.2 Documentation

Mach<sup>Ten</sup> documentation is available in many formats. Online documentation includes the ubiquitous UNIX man pages in `/usr/share/man`, and the GNU info files in `/usr/share/info`. The top level *Documentation* folder on the Mach<sup>Ten</sup> CD-ROM provides extensive documentation in HTML and printable form. The HTML collection includes several GNAT documents, the Lovelace Ada tutorial, many GNU package “info” documents, and the UNIX man pages. The printable collection includes many BSD supplementary papers in PDF format, as well as PostScript versions of the man pages.

### 11.1.3 Program Sources

The sources distributed with Mach<sup>Ten</sup> may be found in the top-level folder of the Mach<sup>Ten</sup> CD-ROM in the *Source\_FFS* fast file system file. The source modules are organized into a number of directories indicative of their origin or use. The default source directory is */base/src*, and is mounted automatically when Mach<sup>Ten</sup> starts up (see section “2.2.2 Accessing Mach<sup>Ten</sup> Sources from the CD-ROM”).

Other directories in */base* are named after target system architectures (e.g., */base/macppc* represents the Power Macintosh). The convention is that object files and executables are built in the target directories, while the source directory remains “pure”.

The BSD and GNU source trees use different *make* programs which have a slightly different *Makefile* syntax. Rather than force one camp into the other, Mach<sup>Ten</sup> provides both *make* programs. One key difference is that BSD *Makefiles* reside in the source directory, while GNU *Makefiles* are created in the object directory by a configuration script. The BSD source directories point to the target object directory with a symbolic link named *obj*. The *obj* symbolic link refers to */var/obj*, which is a symbolic link to the desired target directory in */base*. This allows for a quick change between target architectures, but also requires you to be conscious of which target is currently selected. The GNU *Makefiles* use the *VPATH* variable to point back to the source directory.

When porting or developing new UNIX programs for Mach<sup>Ten</sup>, we recommend that you mimic this file system organization. Following this model may actually ease the job of porting software to the Mach<sup>Ten</sup> system. These strategies make good use of the provided BSD master *Makefiles* in */usr/share/mk*, and ease both developing for multiple targets and sharing your work with other Mach<sup>Ten</sup> users. The symbolic link */base/src/local*, which points to */usr/local/src*, may be used to graft local source packages into this source-object organization.

In section “11.14 Programming Example”, 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<sup>Ten</sup>. This source is in */base/src/bsd4.4/games/rogue*, which further illustrates the structure of the BSD source file hierarchy used in Mach<sup>Ten</sup>.

## 11.2 PEF and XCOFF

Mach<sup>Ten</sup> 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 files, the Mach<sup>Ten</sup> 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 run-time dynamic linking between fragments, and automatic code and data sharing between applications with shared libraries of Code Fragments.

PEF formatted files are developed as the last step in the Mach<sup>Ten</sup> software production process. Internally, the Mach<sup>Ten</sup> development tools rely on an IBM binary executable standard called XCOFF (eXtended Common Object File Format). As the name implies, this file format extends the amount of information kept about an application, including its structure and data. This results in a significantly larger disk file and longer dynamic relocation overhead. For Mach<sup>Ten</sup> debugging purposes, it is possible to retain an XCOFF image of a PEF executable. See section "11.7 Linking Executables" for command line information to retain XCOFF file images.

## 11.3 Shared Libraries

The Mach<sup>Ten</sup> 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. Secondly, 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.

### 11.3.1 Shared Library Production

To produce a shared library with the Mach<sup>Ten</sup> 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 you want exported for linkage with an application. The command line flag “-Xllexportall” exports all global symbols in the shared library. The command line flag “-Xllexport=<filename>” exports the symbols named in the file *filename*. For example, if the file *some\_symbols* contained the following lines:

```
symbA
symbB
symbC
```

by using the command line flag “-Xllexport=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.

### 11.3.2 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 that requested the library for shared library import. Mach<sup>Ten</sup> uses the MacOS Code Fragment loader; the *MachTen Application Libraries* folder, found in the *MachTen Libraries* folder, is in its search path. Thus, the run-time version of shared libraries should be put in the *MachTen Application Libraries* folder for use by the MacOS Code Fragment loader.

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.

### 11.3.3 Run-Time and Compile-Time Naming Conventions

It is important to understand the naming conventions 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:

```
ld -o /hfs/pef/shrllib -xlsharedlibrary -Xlexpall *.o
```

will produce a run-time shared library in the file *shrllib*, and will produce a compile-time shared library in the file */hfs/pef/shrllib.a* that exports all symbols. Note that shared libraries must be created as HFS files because they require the setting of certain Macintosh resources via the MacOS Resource Manager.

## 11.4 Traditional UNIX Libraries

The traditional UNIX libraries (*libc.a*, *libm.a*, etc.) are implemented as shared libraries under Mach<sup>Ten</sup>. These compile-time libraries reside in */usr/macppc/lib*, and the corresponding run-time libraries reside in *System Folder: Extensions: Tenon Application Libraries*.

When an application is created, the compile-time libraries must be used to satisfy symbolic references to functions and data declared in the support libraries. When an application is launched, the shareable run-time libraries are found by the Code Fragment Manager, loaded into memory if they are not already in use, and instantiated for use by the application.

Some compile-time libraries do not have a corresponding run-time library. These libraries are not shareable; hence the loader will statically bind them into the application execution image.

If another Power Macintosh software development environment (MPW, Metrowerks, etc.) is used to build Mach<sup>Ten</sup> applications, the PEF libraries in *System Folder: Extensions: Tenon Application 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.

## 11.5 Header Files

Mach<sup>Ten</sup> includes a number of header files which declare functions, data types, values and macros for supported languages. The C *include* files are ANSI-compliant. Machine-independent files are located in `/usr/include`. Files dependent on a specific target architecture are located in `/usr/<target>/include`. `/usr/macppc/include` contains the *include* files specific to the Power Macintosh. By default, the preprocessor *cpp* scans the target-dependent tree before it scans the generic tree.

### 11.5.1 Pre-Defined Names

The following macro names are used extensively in the Mach<sup>Ten</sup> header and source files to conditionally include or exclude declarations, definitions and code which are or are not applicable to Mach<sup>Ten</sup>:

<code>__MACHTEN__</code>	defined in all Mach <sup>Ten</sup> environments
<code>__MACHTEN_68K__</code>	defined when developing for the Mach <sup>Ten</sup> 68K environment
<code>__MACHTEN_PPC__</code>	defined when developing for the Mach <sup>Ten</sup> PPC environment

Definitions, declarations and code specific to Mach<sup>Ten</sup> should be enclosed as follows:

```
#ifdef __MACHTEN__
    /* Code generic to any MachTen environment */
#endif

#ifdef __MACHTEN_68K__
    /* 68K specific code */
#endif

#ifdef __MACHTEN_PPC__
    /* PPC specific code */
#endif
```



Similarly, definitions, declarations and code which Mach<sup>Ten</sup> does not support should be enclosed as follows:

```
#ifndef __MACHTEN__
    /* Code that is inappropriate for MachTen */
#endif

#ifndef __MACHTEN_68K__
    /* Code that is inappropriate for the 68K */
#endif

#ifndef __MACHTEN_PPC__
    /* Code that is inappropriate for the PPC */
#endif
```

## 11.6 Compiling Sources

The GNU compiler tools are standard in Mach<sup>Ten</sup>. In general, the process of creating an object file consists of:

- preprocessing the source with *cpp*
- performing macro substitutions
- passing the output to the appropriate backend compiler engine which generates assembly code, and
- giving that to *as*, which produces the machine object file in XCOFF format.

By default, the output of each of these stages is written to a temporary disk file. Executable programs are created by loading the appropriate XCOFF files and needed libraries. All or part of this activity may be orchestrated with *gcc*, which executes the appropriate tools in turn.

The following sections present a very brief illustration of the commands for generating object files and executables. Optimization (“-O”) is used in the commands that invoke the backend compiler, causing it to try to produce code that is smaller and faster. Detailed documentation is available in various formats, including man pages, HTML documents, and PDF and PostScript files. The reference materials found in Appendix C of the online Power Mach<sup>Ten</sup> User's Guide may also prove useful.

### 11.6.1 Ada

Compiling and loading Ada source files is similar to that for C and C++; however, it requires a binding step between compilation and loading to check consistency and elaboration order. The backend compiler engine is */usr/macppc/bin/gnat1*.

Using *hello.adb* as an example Ada source, the following command illustrates the typical command for generating a PEF executable:

```
gnatmake hello
```

*gnatmake* is a master program that invokes the tools to perform necessary compilation, binding and linking in the proper order. Alternatively, *gcc* may be used to generate an object file or assembly source as shown below:

```
gcc -c -O hello.adb
gcc -S -O hello.adb
```

These commands generate the files *hello.o* and *hello.s*, respectively. Intermediate files from the compilation tools are discarded.

Detailed documentation may be found in “Appendix A, GNAT for the Macintosh”, in the online Power Mach<sup>Ten</sup> User's Guide.

## 11.6.2 C

C source files are compiled with *gcc*. A symbolic link from *cc* to *gcc* is provided for compatibility with UNIX tradition. The backend compiler engine is */usr/macppc/lib/gcc-lib/powerpc-apple-machten4/2.8.1/cc1*. Files with a “.c” suffix are assumed normal C source files, and those with “.i” are interpreted as previously processed by *cpp* and passed directly to *cc1*. Files ending with “.s” are assembly language files.

Using *hello.c* as an example C source, the following command illustrates creating the PEF executable *hello* from the single source file (intermediate files are discarded):

```
gcc -O -o hello hello.c
```

The object file is sometimes needed, or even the intermediate assembly code or preprocessor output. Again using *hello.c* as our example, the following commands:

```
gcc -O -c hello.c
gcc -O -S hello.c
gcc -E -o hello.i hello.c
```

generate the files *hello.o*, *hello.s* and *hello.i*, respectively.

### 11.6.3 Objective-C

Objective-C source files are compiled with *gcc*. A symbolic link from *cc* to *gcc* is provided for compatibility with UNIX tradition. The backend compiler engine is */usr/macppc/lib/gcc-lib/powerpc-apple-machten4/2.8.1/cc1obj*. Files with a “.m” suffix are assumed Objective-C source files, and those with “.i” are interpreted as previously processed by *cpp* and passed directly to *cc1obj*. Files ending with “.s” are assembly language files.

Using *hello.m* and *Printer.m* as example Objective-C sources, the following command illustrates creating the PEF executable *hello* from the two source files (intermediate files are discarded):

```
gcc -O -o hello hello.m Printer.m -lobjc
```

Sometimes the object file is needed, or even the intermediate assembly code or preprocessor output. Again using *hello.c* as our example, the following commands:

```
gcc -O -c hello.m
gcc -O -S hello.m
gcc -E -o hello.i hello.m
```

generate the files *hello.o*, *hello.s* and *hello.i*, respectively.

### 11.6.4 C++

C++ source files are compiled with *g++*. *g++* is a special wrapper program that executes *gcc* with options appropriate for compiling (and loading) C++ sources. The backend compiler engine is */usr/macppc/lib/gcc-lib/powerpc-apple-machten4/2.8.1/cc1plus*. The file suffixes “.C”, “.cc”, “.cxx”, “.cpp” and “.c++” all denote C++ source files. Files with a “.ii” suffix are interpreted as previously processed by *cpp* and are passed directly to *cc1plus*. Files ending with “.s” are assembly language files.

Using *hello.C* as an example C++ source, the following command illustrates creating the PEF executable *hello* from the single source file (intermediate files are discarded):

```
g++ -O -o hello hello.C
```

Sometimes the object file is needed, or even the intermediate assembly code or preprocessor output. Again using *hello.C* as our example, the following commands:

```
g++ -O -c hello.C
g++ -O -S hello.C
g++ -E -o hello.ii hello.C
```

generate the files *hello.o*, *hello.s* and *hello.ii*, respectively.

## 11.6.5 Fortran

Fortran files are compiled with *g77*. *g77* is a special wrapper program that executes *gcc* with options appropriate for compiling (and loading) Fortran sources. The backend compiler engine is */usr/macppc/bin/f771*. The file suffixes “.F”, “.f” and “.for” all denote Fortran source files.

Using *hello.F* as an example Fortran source, the following command illustrates creating the PEF executable *hello* from the single source file (intermediate files are discarded):

```
g77 -O -o hello hello.F
```

Sometimes the object file is needed, or even the intermediate assembly code or preprocessor output. Again using *hello.F* as our example, the following commands:

```
g77 -O -c hello.F
g77 -O -S hello.F
g77 -E -o hello.i hello.F
```

generate the files *hello.o*, *hello.s* and *hello.i*, respectively.

## 11.6.6 Java

Mach<sup>Ten</sup> comes with Kaffe, a virtual machine to execute Java bytecode. Java source files have the suffix “.java”.

The following environment variables are set for *kaffe*:

```
CLASSPATH ./usr/share/kaffe/classes.zip
KAFFEHOME /usr/share/kaffe
LD_LIBRARY_PATH /usr/lib
```

The following command illustrates compiling Java source:

```
javac hello.java
```

It generates the file *hello.class*, which may be loaded and executed by the following command:

```
kaffe hello
```

Note that the argument to *kaffe* is the class name, not the file name.

## 11.7 Linking Executables

### 11.7.1 ld

MachTen uses *ld* — the link editor (a.k.a. the loader) — to produce executables and shared libraries. It combines object files, searches specified libraries to resolve external function and data references, 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”. 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<sup>Ten</sup> runs in real memory, stack sizing considerations enter into the development of a program. Mach<sup>Ten</sup> provides a “*-Xlstack = <stackbytes>*” parameter that provides the ability to set aside memory for the application stack. This value has a default of 60K, 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 *setstackspace* utility. See section “11.13.1.1 Stack Overrun”.

### 11.7.2 mkpef

The *mkpef* program takes the XCOFF output of the Mach<sup>Ten</sup> loader and produces a PEF executable image. This program is automatically called by the Mach<sup>Ten</sup> loader (in the default case), so typically you will not have to execute this program or build *mkpef* rules into your *Makefile* structure.

## 11.8 To make or pmake

Mach<sup>Ten</sup> provides both the GNU (*/usr/bin/make*) and BSD (*/usr/bin/pmake*) implementations of the *make* utility. Both implementations are in wide use, and certain software packages depend upon unique features of one or the other. (See section “11.1.3 Program Sources” for more information on *make*.)

GNU *Makefiles* are usually self-contained and reside in the object directory. GNU software packages typically require a configuration step which determines the capabilities and features of the host system and customizes the *Makefile* and supporting *include* files accordingly.

BSD *Makefiles* reside in the source directory and usually define a few parameters that direct included master scripts. The master scripts, located in */usr/share/mk*, define rules and default parameters for building and installing executables, libraries, *include* files, man pages and support data. A symbolic link named *obj* is usually created to point to the directory in which object files are built.

## 11.9 Symbol Information

Mach<sup>Ten</sup> application symbol information is obtained by using the *nm* program. *Nm* works with XCOFF files optionally output from the compilation of an application. Use “*-lxcoff*” 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.



## 11.10 Debugging

Since Mach<sup>Ten</sup> uses the Apple-defined PEF format for executable code objects (Code Fragments), there are a number of alternatives for debugging Mach<sup>Ten</sup> applications on the Power Macintosh. The traditional UNIX approach for Mach<sup>Ten</sup> is *gdb*, but Apple and other third party debuggers are also available and will work with Mach<sup>Ten</sup> PEF Code Fragments.

### 11.10.1 Debugging using *gdb*

Mach<sup>Ten</sup> applications can be debugged using the Mach<sup>Ten</sup> *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:

```
gcc -g -o hello hello.c
```

will produce the PEF output file *hello* and the XCOFF output file named *hello.xcoff*. Note that the optimization flag “-O” has been left out to avoid confusing matches between source code lines and the generated assembly code.

The Mach<sup>Ten</sup> 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, standard UNIX *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* man page.

## 11.10.2 Macintosh Debugging Tools

First, determine that a debugger is not already installed on your system. If the message “Debugger Installed” is printed when the system is booting, it typically 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.

### 11.10.2.1 MacsBug

MacsBug<sup>†</sup> is a Macintosh debugger that offers low-level debugging support on Power Macintoshes. MacsBug works with Mach<sup>Ten</sup>, providing assembly-level debugging, break points and PowerPC exception handling. It interprets traceback data attached to each Mach<sup>Ten</sup>-generated subroutine showing routine name and parameter information.

If you do not already have a debugger installed, it is recommended that you install MacsBug in your System Folder. It is located in *Utilities:Development Utilities:Debuggers:MacsBug* on the Mach<sup>Ten</sup> CD-ROM. It will provide you with a backup if one of your applications crashes.

By using MacsBug, you can display memory as PowerPC assembler code (use the command “`ilp <addr>`”). 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 database address from the MacsBug “`CFM`” command and the offset information from the “`nm`” command provide an absolute memory location for a variable.

---

<sup>†</sup> [ftp://ftp.support.apple.com/pub/apple\\_sw\\_updates/us/mac/utlis](ftp://ftp.support.apple.com/pub/apple_sw_updates/us/mac/utlis)

### 11.10.2.2 Macintosh Debugger for PowerPC

The Macintosh Debugger for PowerPC is Apple's symbolic debugging tool for Power Macintosh. Version 2.0.d3 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<sup>Ten</sup> application uses a Thread Manager stack).

### 11.10.2.3 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<sup>Ten</sup> applications if the applications are also being developed using the other Code Warrior tools.

### 11.10.3 Environment Variables for Debugging and Monitoring

Each Mach<sup>Ten</sup> application is run with certain pre-set information known as the “environment”. The environment is a collection of variables stored as character strings. Each string specifies the name and value of one variable. Many environment variables are set during login and when starting a new shell.

For *cs*h and *tc*sh, the commands to set and unset environment variables are:

```
setenv <variable> <value>
unsetenv <variable>
```

For *sh* and *ba*sh, the commands to set and unset environment variables are:

```
export <variable=value>
unset <variable>
```

Consult the *man* page for your shell of choice for the complete details on setting and unsetting environment variables, and the special shell script file(s) that are processed during startup.

The environment variables `DEBUGGERFIRST`, `STACKCHK` and `MEMSTATS` direct Mach<sup>Ten</sup> to perform special debugging or monitoring actions when they are set to a non-zero value (typically “1”). Environment variables set in a given process are only propagated to child processes. It is often convenient to establish one terminal window with the desired “debugging environment”, and to have additional terminal windows for editing and other activities. Unset the environment variables to discontinue their service.

### 11.10.3.1 DEBUGGERFIRST

When set, the environment variable `DEBUGGERFIRST` causes Mach<sup>Ten</sup> to execute a “`Debugger()`” MacOS system call immediately before entering the “`main`” function of the application. This allows you to set initial breakpoints and step through initial startup actions such as C++ constructor calls. For this to work, you must have previously installed a low-level debugger, such as MacsBug. By default, Mach<sup>Ten</sup> will trap this “`Debugger`” call. To allow the “`Debugger`” call to pass through to the MacOS and MacsBug, or other debuggers, rename the Mach<sup>Ten</sup> application to *MachTen.noexcept* and restart Mach<sup>Ten</sup>.

### 11.10.3.2 STACKCHK

When set, the environment variable `STACKCHK` causes Mach<sup>Ten</sup> to report (on the console terminal) the amount of stack used when a program exits. This is done by zeroing the entire stack prior to executing the program, and scanning for the “highest” non-zero stack entry when the program quits. Note that stacks grow downward from a high address to a low address.

### 11.10.3.3 MEMSTATS

When set, the environment variable `MEMSTATS` causes Mach<sup>Ten</sup> to report memory usage statistics when a program exits. The report includes the maximum amount of memory allocated and the number of “buckets” in use and available. A bucket is simply a buffer whose size is a power of 2. The report also tallies the number of calls to *malloc*, *free*, *realloc*, *alloca*, and the *sbrk* and *sfree* system calls. This information may help to identify inefficient memory usage.

## 11.11 Making Macintosh Applications

One of the benefits of producing PEF formatted binary images is that the Mach<sup>Ten</sup> 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 also specify a Macintosh-specific startup routine providing for proper startup data definition and proper program exit, and follow Apple system programming rules (e.g., *WaitNextEvent*, *MoreMasters*, etc.). In addition, the software must not make UNIX system calls or calls to UNIX libraries.<sup>†</sup>

### 11.11.1 Macintosh OS Header Definition Files

Macintosh applications must be built with Macintosh interface header definition files for data/code typing and parameter access. Mach<sup>Ten</sup> includes Apple's C include files in the directory */usr/include/MacOS*.

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<sup>Ten</sup> development applications requires that these files have a Mach<sup>Ten</sup> 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 onto the Mach<sup>Ten</sup> *Unix<->Text* application. Drop the group on the application and let it change each of the files. The Mach<sup>Ten</sup> *Unix<->Text* application is contained in the *Utilities* directory of the Mach<sup>Ten</sup> 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<sup>Ten</sup> and traditional Macintosh development applications.

---

<sup>†</sup> If you want to create a "pure" Macintosh application that uses UNIX system calls or standard UNIX libraries, please contact Tenon. Licensing a standard library package from Tenon that you can incorporate into your "Macintosh application" will let you develop hybrid applications.

## 11.11.2 Macintosh OS Interface Libraries

To include a standard Apple interface library with Mach<sup>Ten</sup> 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. The Macintosh creator must be modified to incorporate one of these libraries on a Mach<sup>Ten</sup> command line. Generally this has no effect on access to the library by the native development software. Modifying a library creator to "MUMM" will allow Mach<sup>Ten</sup> to access the file as a binary file. The file should already have a Macintosh type identification of "shlb", which is used by Mach<sup>Ten</sup> 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 *finderinfo(1)* or *ResEdit* and selecting the "Get Info" function from the File menu. Creator information can be modified under *ResEdit* with traditional Macintosh "point-and-click" methods.

Do not to make a copy of one of the libraries using Mach<sup>Ten</sup> applications until the creator has been changed to "MUMM". Access to a library not created by "MUMM" will automatically create an AppleSingle version of the copied data and make it unusable by Mach<sup>Ten</sup> 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<sup>Ten</sup> development tools will place a reference in the resulting PEF output to a Code Fragment with the same name as 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<sup>Ten</sup> 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<sup>Ten</sup> 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:=fragmentname> -creator <xxxx> -type  
shlb <inputfile.xcoff> <outputfile>
```

where “xxxx” is a four-character Macintosh creator.

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 “shlb” must also be specified for a shared library or a type “APPL” for an application. Creator types should be registered with Apple Computer at <http://www.devworld.apple.com/dev/cftype>.

### 11.11.3 Macintosh Application Startup Routine

Standard Mach<sup>Ten</sup> applications use a startup routine that sets UNIX parameters and environment variables. The startup routine then calls a subroutine called “main” and, on return from “main”, calls the standard UNIX system call “exit”. 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<sup>Ten</sup> distribution under the file name */usr/lib/MacOS/Mac\_start.o*. A slightly modified version that enters the Macintosh debugger first is in */usr/lib/MacOS/Mac\_debug.o*.



### 11.11.4 Macintosh Application Construction

The newly modified interface libraries and header definition files must be specified during the compilation and linkage phases of the Mach<sup>Ten</sup> code production process. This is specified using standard UNIX methods. Access to a directory of header definition files is specified using the “-I” parameter on a UNIX command line. The command line:

```
gcc -O -I/usr/include/MacOS -Dpascal="" -c hello.c
```

will define (with “-D”) “pascal” to be nothing for MacOS compatibility, and will search the */usr/include/MacOS* directory for *include* files. It will also use “-O” optimization as usual for a faster and more compact binary.

The command line:

```
ld -o /hfs/APPL/hello -Xlcreator 'MISC' -Xltype APPL \  
/usr/macppc/lib/MacOS/Mac_start.o \  
hello.o \  
/usr/macppc/lib/MacOS/InterfaceLib.a
```

will create an output file *hello* in the */hfs/APPL* folder. Note that the application must be created as an HFS file because it requires the setting of certain Macintosh resources via the MacOS Resource Manager.

The *ld* command includes the Macintosh application initialization code from *Mac\_start.o*, as well as the graphic user interface code from *InterfaceLib.a*. You must specify the full name to both of these supplementary pieces of code. Since *InterfaceLib.a* is a shared library, exported symbols from the library will be treated as dynamic linkages to be satisfied during the run-time execution of the application. The “Xlcreator” and “Xltype” parameters specify a “MISC” creator and a standard Macintosh application type. The resulting “point-and-click” Macintosh program can be executed independently of Mach<sup>Ten</sup>.

## 11.12 Cross-Development Tools and Targets

The *macppc* component of the target pathnames used in the BSD and GNU *Makefiles* is for the potential use of Mach<sup>Ten</sup> in a cross-development environment (i.e., */base/mac68k/bsd4.4* could be the target directory for building 68K versions of the BSD applications). The locations of the development tools are also dictated by the cross-development capabilities of Mach<sup>Ten</sup>.

“machine-specific” development tools	<i>/usr/macppc/bin</i>
libraries	<i>/usr/macppc/lib</i>
header files	<i>/usr/macppc/include</i>
non-“machine-specific” development tools	<i>/usr/bin, /usr/lib and /usr/include</i>

Links to the machine-specific development tools are set up in the corresponding directories.

### 11.12.1 Default Mach<sup>Ten</sup> Environment

Selection of the target in a cross-development environment is controlled by the `PATH` variable and other environment variables and symbolic links. (See section “5.4 Tailoring Your UNIX Environment” for information on environment variables.)

Currently Mach<sup>Ten</sup> provides support for PowerPC targets; thus, this environment is established as the default.

## 11.13 Porting Software to Mach<sup>Ten</sup>

Where possible, every effort has been made in Mach<sup>Ten</sup> to provide as complete a BSD UNIX environment as possible. However, due to the lack of memory protection in MacOS, Mach<sup>Ten</sup> is a “real memory” implementation of UNIX. Therefore, Mach<sup>Ten</sup> 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 that the programs that do depend on virtual memory can be easily diagnosed as such, and modified to run under Mach<sup>Ten</sup>.

### 11.13.1 Real Memory Issues

#### 11.13.1.1 Stack Overrun

Mach<sup>Ten</sup> starts each process with a fixed size stack. If a subroutine defines large amounts 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 automatic variables are to redefine them as static, or to allocate them with *malloc* or *alloca*. The default process stack size is 36K bytes. A larger process stack size may be specified using the *setstackspace* utility. A larger stack may also be specified when a program is built with the “-Xlstack” loader option (see section “11.7 Linking Executables”).

Stack overrun is probably the number one real memory issue and should be investigated first when a program is sporadically failing within Mach<sup>Ten</sup>. For example, an application may build and run fine when filtering a single file, but crash when filtering a large amount of files.

### 11.13.1.2 Allocating Memory in Mach<sup>Ten</sup>

Mach<sup>Ten</sup> 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. Secondary memory allocations occur when a running application requests memory, which can obviously occur only after an application has been properly loaded by the CFM.

The CFM uses two areas of memory to store Code Fragments — Mach<sup>Ten</sup> application memory (Mach<sup>Ten</sup> heap), and system memory (system heap). Mach<sup>Ten</sup> application data fragments are allocated from Mach<sup>Ten</sup> application memory. Mach<sup>Ten</sup> 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 application may make subsequent requests for memory (secondary memory allocations). These requests are allocated from Macintosh system memory.

A dynamic bar chart of allocated system memory, as well as Mach<sup>Ten</sup> and other application memory, can be displayed by selecting the “About This Computer” entry in the Apple menu when Finder is the front application.

Sometimes, due to heavy system loads, either Mach<sup>Ten</sup> memory or system memory allocations will be completely exhausted. By reallocating the size of the Mach<sup>Ten</sup> memory space, it is possible to change the relative sizes of the two allocations. See section “11.13.1.4 Setting the Mach<sup>Ten</sup> Heap Size” for directions on how to increase the Mach<sup>Ten</sup> memory heap.

Mach<sup>Ten</sup> typically fails to allocate memory in two ways — the CFM fails to find sufficient memory when loading an application (static allocation), or the application itself fails during a secondary memory request (dynamic allocation). Both failures print an error message. When the CFM fails to find enough memory to load an application's data, the error message is “CFM error -2810”. Getting this error means that the Mach<sup>Ten</sup> memory space is fully allocated. One solution is to quit Mach<sup>Ten</sup>, increase the Mach<sup>Ten</sup> heap size, and re-launch Mach<sup>Ten</sup>.

When CFM fails to find enough memory to load an application's code, the error message is "Frag load err -108". Getting this error means that the system memory space is fully allocated. One solution is to quit Mach<sup>Ten</sup>, decrease the Mach<sup>Ten</sup> heap size, and re-launch Mach<sup>Ten</sup>. It is also a good idea to quit other applications and increase the Macintosh's virtual memory.

A secondary memory allocation is a request by the running application to allocate memory. This is typically achieved via the "alloc()" and "malloc()" family of library calls. Secondary memory allocations are completely independent of the CFM memory allocations, and can occur only after an application has been successfully loaded by the CFM. Secondary memory allocations are allocated from Macintosh system memory.

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. Possible solutions include quitting any Macintosh applications that might also be using system memory, closing any unused Finder windows, or ultimately quitting Mach<sup>Ten</sup> and reducing its heap allocation size. Note that by reducing the size of the Mach<sup>Ten</sup> heap, the size of the allocatable system memory is increased, so it is possible to iterate several times around increasing or decreasing the Mach<sup>Ten</sup> heap size to best suit a particular system's memory profile.

### 11.13.1.3 Calculating Memory Requirements

The Code Fragment Loader loads the code (or executable) portion of Mach<sup>Ten</sup> applications in the system heap, and the data portion of Mach<sup>Ten</sup> applications in the Mach<sup>Ten</sup> heap. Therefore, a running Mach<sup>Ten</sup> application consumes space in both the system and Mach<sup>Ten</sup> heaps, and it is necessary to have space available in both of these heaps in order to successfully run Mach<sup>Ten</sup> applications.

The "About This Computer" 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 system and the Mach<sup>Ten</sup> heaps. However, it is not easy to predict how much more memory a particular Mach<sup>Ten</sup> application will require. Mach<sup>Ten</sup> libraries (*libc*, *libm*, etc.) are shared libraries. Mach<sup>Ten</sup> applications that reference these libraries share a single instantiation of the library's code. The library's data segments are not shared. Each Mach<sup>Ten</sup> application allocates a

private copy of the data segments for each shared library it references. This dynamic sharing of the Mach<sup>Ten</sup> libraries, combined with the stack and memory allocations internal to Mach<sup>Ten</sup> programs, make it difficult to determine how much additional memory a Mach<sup>Ten</sup> program will consume from the Mach<sup>Ten</sup> and system heaps.

Examining the “About This Computer” display both before and after launching a Mach<sup>Ten</sup> application provides a way to approximate memory requirements. Using MacsBug (or another debugger) to dump the exact heap statistics provides a more precise means for calculating memory usage.

#### 11.13.1.4 Setting the Mach<sup>Ten</sup> Heap Size

Setting the size of the Mach<sup>Ten</sup> heap varies the sizes of the code and data allocation pools. Increasing the size of the Mach<sup>Ten</sup> heap generally increases the size of the data allocation memory area at the expense of decreasing the code allocation size. To set the Mach<sup>Ten</sup> heap size, first confirm that Mach<sup>Ten</sup> is not running. Next, find the Mach<sup>Ten</sup> application icon and highlight it with a single mouse click. Then pull down and select “Get Info” from the Finder File menu. This will display information about the Mach<sup>Ten</sup> application. The lower right-hand corner of the display will contain a “Suggested Size” setting for Mach<sup>Ten</sup>. Select the box and raise or lower the setting according to your system's needs. Close the display by clicking the close box. Restart Mach<sup>Ten</sup>. Verify the new size by opening the dynamic memory display by selecting “About This Computer” in the Finder's Apple menu.

### 11.13.1.5 Problem Areas

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:

etext, edata, end	In a virtual memory environment, a program can make assumptions about the relative locations of the program's <i>text</i> , <i>data</i> and <i>bss</i> 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.
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 <sup>Ten</sup> . In Mach <sup>Ten</sup> , "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. Memory is a scarce resource in a real memory implementation, and memory requests are typically rounded to a much smaller size (multiples of 16 or 32 bytes). Accidentally exceeding this real memory boundary will most certainly cause problems to the actual owner of this space.
fragmentation	In a real memory situation, it is possible 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 <i>realloc()</i> call, with successively larger sizes. Quitting and restarting Mach <sup>Ten</sup> will solve this.

out of memory during compile	<p>Both the C compiler (<i>gcc</i> and <i>cc1</i>) and the <i>make</i> program require a lot of memory to run. If you are compiling large files and have an elaborate program generation environment built up where a <i>Makefile</i> causes multiple copies of <i>make</i> to run at the same time, this will require large amounts of memory and may cause either the C compiler or <i>make</i> to fail during some memory allocation request, and thereby aborting the program generation. A potential solution to this problem is to use some of <i>make</i>'s more sophisticated features, such as <i>VPATH</i>, and to reorganize the program generation process to be completely controlled from one or two <i>make</i> invocations.</p>
sticky bits	<p>The <i>cc1</i>, <i>cc1plus</i>, <i>f771</i>, <i>cclobs</i> and <i>gnat1</i> compilers are installed with the "sticky bit" set. Once these compilers are used, they remain in memory until Mach<sup>Ten</sup> quits. This speeds execution, but may reserve too much memory if you are using multiple languages and have limited memory. Removing the sticky bit will cause the programs to be loaded into memory each time they are used. This may be done by typing the following command (as <i>root</i>):</p> <pre>chmod 555 /usr/macppc/bin/&lt;filename&gt;</pre>



## 11.14 Programming Example

### 11.14.1 Rogue

BSD 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<sup>Ten</sup>.

### 11.14.2 Building the Executable

First, it is necessary to build the executable binary image for the *rogue* program. The *Makefile* has already been set up for the Mach<sup>Ten</sup> 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
pmake objdir
pmake
```

The first *pmake* command creates the directory that the symbolic link *obj* points to, if it does not already exist. In this example, it is */base/macppc/bsd4.4/games/rogue*. This is the directory in which the objects and executable will be generated.

### 11.14.3 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 made only if the compile options include the “-DDEBUG” flag (pmake 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<sup>Ten</sup> to quit executing. In order to pass this “Debugger( )” call through to the MacOS, rename the Mach<sup>Ten</sup> application to *MachTennoexcept* and restart Mach<sup>Ten</sup>.

## 12.0 The X Window System

The X Window System®, also known simply as 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. The client and server communicate with one another using the “X Protocol”.

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<sup>Ten</sup> X Window software supports the X client/server model under Mach<sup>Ten</sup> on a Macintosh. Using the Mach<sup>Ten</sup> X Window software, the Macintosh display is transformed into a high-performance X server. Mach<sup>Ten</sup> can be used to create and run X client programs. Simultaneous operation of Macintosh applications with X clients is also possible by using Finder 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<sup>Ten</sup> X Window software. Related documentation on X is listed in “Appendix C, Suggested Reading for Programming Languages”, in the online Power Mach<sup>Ten</sup> User's Guide.

The following section introduces the X environment under Mach<sup>Ten</sup>. It assumes a completed installation of Mach<sup>Ten</sup> and the X Window software on your Macintosh. The software installation instructions are in section “2.0 Installing Mach<sup>Ten</sup>”. An overview of the X Window software components appears in section “12.3 Mach<sup>Ten</sup> X Window Software Overview”.

## 12.1 The X Desktop

The Mach<sup>Ten</sup> X Window server runs entirely within a window in the Mach<sup>Ten</sup> windows environment. For those unfamiliar with this environment, refer to section “4.2 Mach<sup>Ten</sup> 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<sup>Ten</sup> 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<sup>Ten</sup> terminal and other Macintosh application windows will appear on the Macintosh desktop.

### 12.1.1 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:

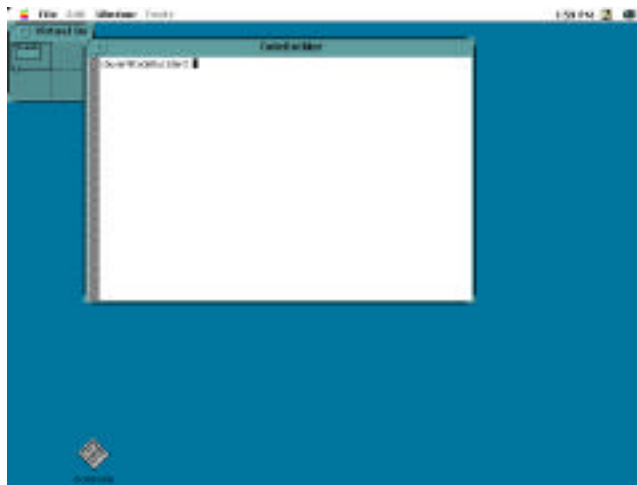


Figure 30. The X Desktop

The Mach<sup>Ten</sup> *wind(8)* process MUST be running in order to start the X server. Use the command “`ps -ax`” to determine if *wind* is running (see “Figure 24. Using *ps* to Show What is Running”). To execute *wind*, type “*wind*” at the prompt. See also “12.1.3 Automatic Launch of the X Server”.

The X server desktop window differs from a Mach<sup>Ten</sup> 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<sup>Ten</sup> 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 30. The X Desktop”, the *xconsole*, *xterm* and *olvwm* applications are running. X client startup is discussed in section “12.2.1 Starting Clients”.

## 12.1.2 The Menu Bar

The Mach<sup>Ten</sup> menu bar appears above the X server desktop (“Figure 30. The X Desktop”) and controls the X server's visibility on the desktop relative to other Mach<sup>Ten</sup> terminal windows and Macintosh applications you may be running. Unlike a Mach<sup>Ten</sup> terminal window, <Command> key shortcut menu selections (the “hot” keys) are disabled when the X desktop is visible, since the <Command> key is meaningful as an X <Meta> key within the X environment.

### 12.1.2.1 The File Menu

The File menu lets you create Mach<sup>Ten</sup> terminal windows within the Mach<sup>Ten</sup> windows environment. The File menu is also used to quit the X Window server environment.

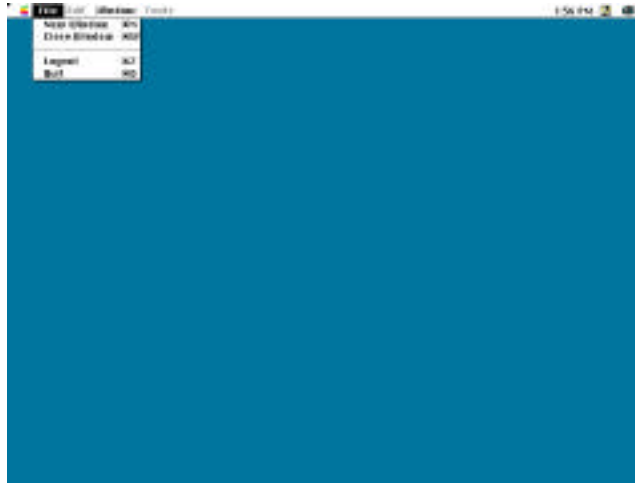


Figure 31. The File Menu

New Window	Create a new Mach <sup>Ten</sup> 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 <sup>Ten</sup> windows, including the X server, and return to the login console.
Quit	Quit the Mach <sup>Ten</sup> application.

### 12.1.2.2 The Window Menu

The Window menu contains a hierarchical “Order” sub-menu item, a “Save Window Environment” item, and a list of Mach<sup>Ten</sup> terminal windows, including the X server. To access a sub-menu item, slide the mouse cursor over the item and to the right.

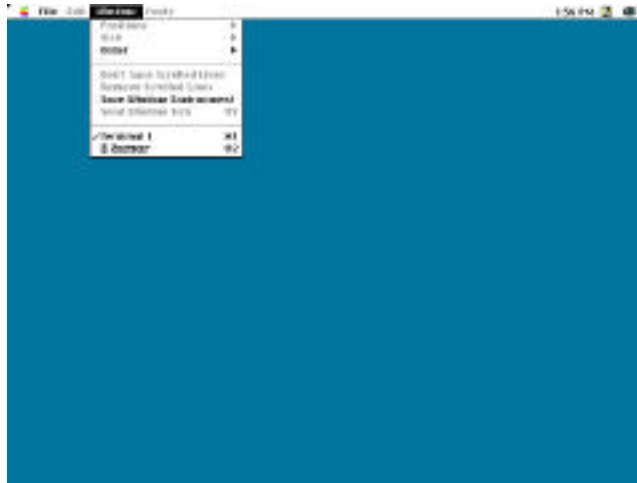


Figure 32. The Window Menu

Save Window Environment	Save a description of the sizes and locations of the current Mach <sup>Ten</sup> 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 <sup>Ten</sup> windows environment.
The Window List	Selecting a Mach <sup>Ten</sup> 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<sup>Ten</sup> windows when the X server is active.

### 12.1.2.2.1 The Order Sub-Menu

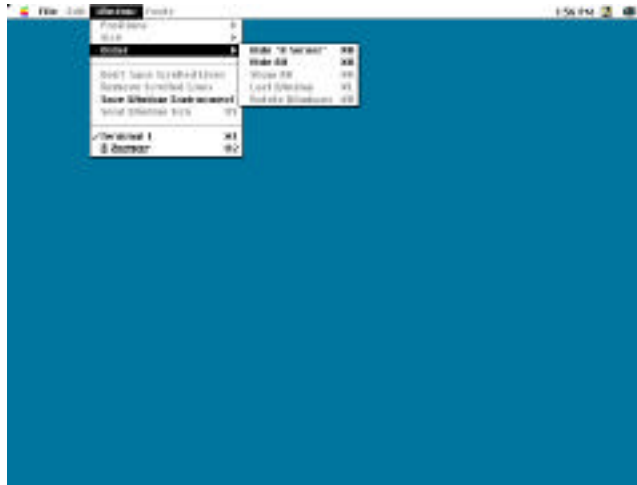


Figure 33. The Order Sub-Menu

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 <sup>Ten</sup> terminal windows invisible.
Show All	Make the X server visible.



### 12.1.3 Automatic Launch of the X Server

The X server may be started automatically the next time you log into the Mach<sup>Ten</sup> windows environment by selecting “Save Window Environment” from the Window menu. This adds a single character lowercase “x” appearing on a line by itself in your Mach<sup>Ten</sup> Window environment startup file (*.windrc* by default) that indicates the X server will start up when the Mach<sup>Ten</sup> windows environment is entered.

If you close all Mach<sup>Ten</sup> 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<sup>Ten</sup> terminal window in the new environment.

### 12.1.4 Quitting the X Server

To quit the X server and remove the X desktop, pull down the File menu and select “Close Window”.

### 12.1.5 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 pressing the mouse button while dragging the cursor along the top edge of your X desktop.

Refer to the *-fs* X server startup option in section “12.2.3 The X Server Program” for more information on running the X server without a Macintosh menu bar.

#### 12.1.5.1 Menu Bar Shortcuts

You can perform many of the menu bar functions described in this section using the Mach<sup>Ten</sup> *xtmenu* 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<sup>Ten</sup>. 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 "Order" item under the Window menu.
-s	When Mach <sup>Ten</sup> 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 startup sequence.
-q	When Mach <sup>Ten</sup> is the foreground application, this option closes all X client connections, terminates the X server, and removes the X desktop from the Mach <sup>Ten</sup> 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 *olvwmm* using the <Shift> and <Escape> keys by modifying the *\$HOME/.olvwmrc* file as follows:

```
Escape + Shift {
    Execute: "/usr/bin/X11/xtmenu -h"
}
```

Window managers (with an emphasis on *olvwmm*) are discussed in section "12.2.2 The Window Manager Client". Refer also to the *olvwmm*, *olvwmrc*, *afterstep*, and *twm* online manual pages for more information on the *.olvwmrc*, *.steprc* and *.twmrc* files.

## 12.2 Administering the X Window Software Environment

The Mach<sup>Ten</sup> X Window software is a very flexible software package that can be tailored to your individual requirements. This section describes how to 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<sup>Ten</sup> 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.

### 12.2.1 Starting Clients

When you type <Command-T> or enter the command:

```
xtmenu -s
```

the *xinit* X initialization program is launched by the Mach<sup>Ten</sup> kernel. *xinit* is responsible for starting *XMachTen*, the X server program, and a default set of X clients to display on your X desktop from a startup script.

*xinit* may only be run by the Mach<sup>Ten</sup> kernel.

#### 12.2.1.1 The Startup Script

The initial X clients you choose to run on your X desktop are controlled by a shell script file in your Mach<sup>Ten</sup> 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 startup script `/usr/lib/X11/xinit/xinitrc` is shown below:

```
#!/bin/sh

userresources=$HOME/.Xresources
usermodmap=$HOME/.Xmodmap
sysresources=/usr/X11/lib/X11/xinit/.Xresources
sysmodmap=/usr/X11/lib/X11/xinit/.Xmodmap

# merge in defaults and keymaps

if [ -f $sysresources ]; then
    xrdp -merge $sysresources
fi

if [ -f $sysmodmap ]; then
    xmodmap $sysmodmap
fi

if [ -f $userresources ]; then
    xrdp -merge $userresources
fi

if [ -f $usermodmap ]; then
    xmodmap $usermodmap
fi

# Set the root window background

xsetroot -grey

# Start a console terminal

xconsole -iconic &

# Start the User's Custom Client Scripts that are selected for
autolaunch
./client_autolaunch

# Start some programs
#xterm -geometry 80x24+100+20 -ls -n `hostname` -T `hostname` &
#xclock -geometry 50x50+1-1 &

# Start a window manager
#WINDOW_MANAGER=olvwm
#WINDOW_MANAGER=twm
#WINDOW_MANAGER=mwm
WINDOW_MANAGER=afterstep
```

```

if [ -x /usr/X11/bin/$WINDOW_MGR ]; then
  if [ $WINDOW_MANAGER = "afterstep" ]; then
    LOW_COLOR=`xdpyinfo | grep "depth of root window:8 planes"`
    if [ -n "$LOW_COLOR" ]; then
      if [ -r "$HOME"/.steprc.8bit ]; then exec afterstep -f "\
$HOME"/.steprc.8bit
      else exec afterstep -f\
/usr/X11/lib/X11/afterstep/system .steprc.8bit
    fi
  fi
fi
exec $WINDOW_MANAGER
else
  exec /usr/X11/bin/xterm
fi

```

The startup script begins by calling the *xrdb* client which reads in any system-wide resources defined in the file */usr/X11/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 section “12.2.1.2 Resources — X Application Preferences”.

The script then runs the *xmodmap* client to load a custom keyboard mapping file from */usr/X11/lib/X11/xinit/Xmodmap*, if present. You will have no need to change your keyboard mapping if you are using a U.S. keyboard, as 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/X11/lib/X11/xinit*. Details on using the international keyboard map files are found in section “12.2.3.3 Keyboard Mapping”.

The startup script runs *xrdb* and *xmodmap* again using the *.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 startup script uses *xsetroot* to change the background, or root window, to a grey color. Choose any color from the list of colors in */usr/X11/lib/X11/rgb.txt*.

Next, *.xinitrc* starts a console X window as an icon on the X desktop. Mach<sup>Ten</sup> console output is redirected to the *xconsole* window and is visible when the icon is “de-iconified”, or opened with a double-click of the mouse button.

`.xinitrc` then starts the X applications and, by default, an `xterm` 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` client is not run by default because of the comment (“#”) 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 startup script launches a special X application known as the “window manager” that allows you to move, resize and iconify your X application windows. The AfterStep Window Manager (`afterstep`), based on NeXTSTEP™, is launched by default. Also available in Mach<sup>Ten</sup> is the OpenLook Virtual Window Manager (`olvwm`), the Tab Window Manager (`twm`) and the Motif Window Manager (`mwm`). The operation of the OpenLook Virtual Window Manager is described in detail in section “12.2.2 The Window Manager Client”.

The startup script you create in `.xinitrc` in your home directory takes precedence over the default system startup script.

### 12.2.1.2 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<sup>Ten</sup> allows multiple simultaneous users, X application preferences can also be specified for each individual user.

**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` 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 to have a blue background for programming C, and a red background for Ada. Your resource file may contain the following resources:

```
xterm-C*background:blue  
xterm-Ada*background:red
```

When you start the `xterm` client for programming C or Ada, use the “-name” option to match the instance name used in the resource:

```
xterm -name xterm-C & (xterm started for C)  
xterm -name xterm-Ada & (xterm started for Ada)
```

“Instance name” 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 (*XTerm*, for example), 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/X11/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* sub-directory under the */tmp* directory.

- **\$HOME/.Xdefaults**

If the X server database (described in the next section) 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 can be stored in a database residing on the X server using the *xrdb* program. This is typically done when the server starts up in the *.xinitrc* file in the user's home directory. If the file `/usr/X11/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. 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 AppleShare-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., an 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/X11/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 in doubt as to which resources an X application will use when it is launched, use the *appres* 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.

## 12.2.2 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.

### 12.2.2.1 Starting the Window Manager

Your default client startup script (*.xinitrc* in your home directory or */usr/X11/lib/X11/xinit/xinitrc*) 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.

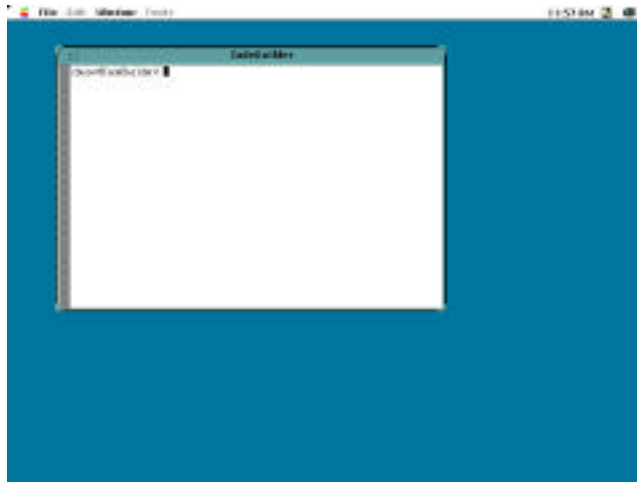


Figure 34. The OpenLook Environment

### 12.2.2.2 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” an item, it simply means move the mouse until the pointer on your screen is positioned over the item.

**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. When you point to the corner of a window, the pointer changes to a circle shape.

**Clicking, Double-Clicking and Dragging.** The following terminology is used to refer to actions involving mouse buttons:

- “Click” means press and release a button without moving the pointer.
- “Double-click” means click a button twice in rapid succession.
- “Drag” means press and hold a button while moving the pointer.

### 12.2.2.3 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.

When a window is active, the window frame changes appearance. Now, when you press on the keyboard and the pointer is in that window, the characters appear on the command line in the active window.

There are two different models of handling the mouse focus. The first, “Focus Follows Mouse”, is the default behavior.

This behavior can be toggled by selecting “Focus” from the OpenLook “Workspace” menu.

Selecting this option changes the focus model to “Click To Focus”. With this model, you must position the mouse over the window and then click the select button (the select button is your mouse button or the left button on a three-button mouse) to focus.

If no window is active, everything you type will be lost.

### 12.2.2.4 Moving a Window

By 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 — Grab 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 word “xterm” appears. When properly positioned, the pointer is an arrowhead as shown in “Figure 35. Grabbing the Title Bar”.



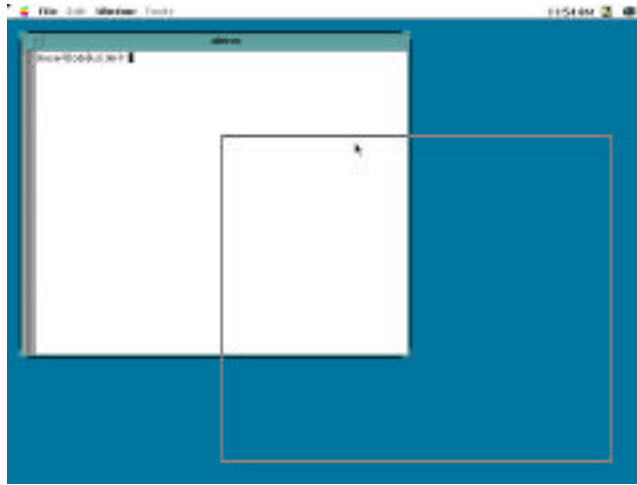
**Figure 35. Grabbing the Title Bar**

When the pointer is positioned, press and hold the select button.

**Step 2 — Drag the Window to a New Location**

While 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 36. Repositioning the Window”. The outline shows you where the window will be moved when you release the mouse button.



**Figure 36. Repositioning the Window**

**Step 3 — Release 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.

### 12.2.2.5 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 — Grab the Frame with the Pointer***

Where you grab the window's frame determines how the window will be resized. See "Table 7: Places to Grab on the Window Frame".

For this exercise, point to the lower right corner of the window frame. When positioned correctly, the pointer's shape changes to a circle around the corner of the frame. Grab the frame by pressing and holding down the select button.

**Table 7: Places to Grab on the Window Frame**

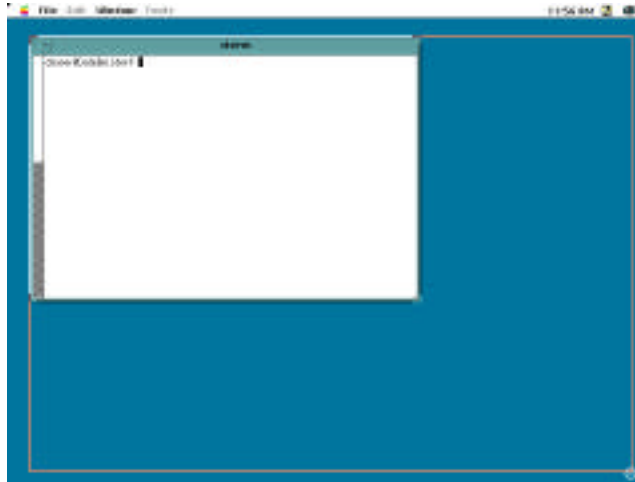
<b>If you want to stretch or shrink the window diagonally from the ...</b>	<b>Point to ...</b>
bottom left	frame's lower left corner
top left	frame's upper left corner
top right	frame's upper right corner
bottom right	frame's lower right corner



**Step 2 — Enlarge or Shrink 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.

Now, stretch the window until it reaches the lower right corner of the root window as shown in “Figure 37. A Window and Its Elastic Outline”.



**Figure 37. A Window and Its Elastic Outline**

**Step 3 — Release the Frame**

When the elastic outline is the correct size, release the mouse button. The screen is redrawn with the window filling the outline.

### 12.2.2.6 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 — Locate 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 38. The Title Bar Minimize Button”.



Figure 38. The Title Bar Minimize Button

#### **Step 2 — Press the Minimize Button**

Press the minimize button by clicking the select button. The window changes into an icon. *olwmm* displays icons in a row along the bottom of the root window, starting on the left.



Figure 39. The OpenLook Icon

### 12.2.2.7 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. While 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.

### 12.2.2.8 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 (i.e., press the 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 40."

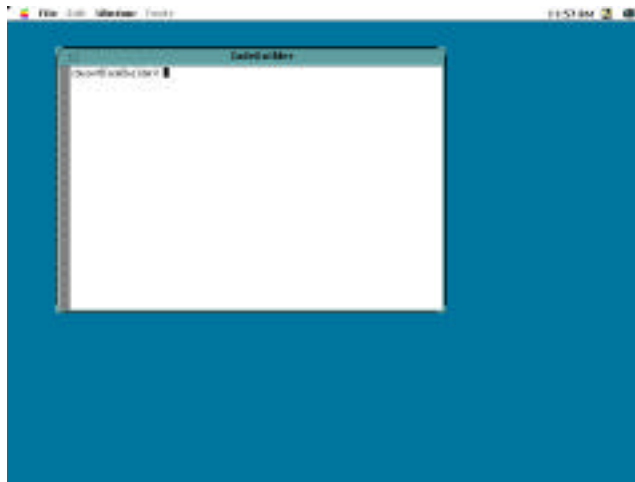


Figure 40. The Restored Window

### 12.2.2.9 Displaying a Window Menu and Making Selections

Each window has a Window menu that contains functions for controlling the window.

#### *Step 1 — Display the Window Menu*

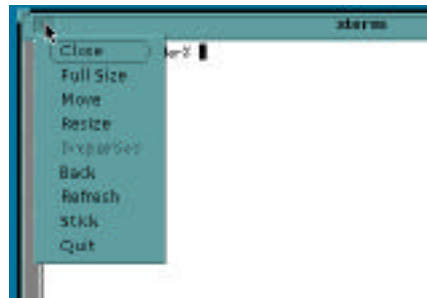
The Window menu is attached to the window frame. Use the menu button in the upper left corner of each window to display it.

To display the Window menu, point to the menu button and then press and hold the third button.<sup>†</sup> The menu is displayed as long as you hold the button down — don't release the button yet. See "Figure 41. The OpenLook Window Menu".

#### *Step 2 — Choose 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 "Full Size" function. Release the mouse button.

The "Full Size" function causes the window to expand to fill the entire screen vertically.



**Figure 41. The OpenLook Window Menu**

---

<sup>†</sup> Please refer to section "12.2.3.2 Mouse Button Mapping", *Three Button Mouse Configuration*.

**Step 3 — Restore 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 Size” selection. Release the mouse button. The window is restored to its former size and location. “Table 8: Functions in the Window Menu” lists the Window menu functions.

**12.2.2.10 Summary of Window Menu Functions****Table 8: Functions in the Window Menu**

To do this ...	Choose ...
Restore a window from an icon or after maximizing.	Restore Size
Change the location of a window.	Move
Change the size of a window.	Resize
Shrink a window to its icon representation.	Close
Enlarge a window to be as tall as the root window.	Full Size
Send a window to the back or bottom of the window stack, the position closest to the root window.	Back
Immediately stop a window and make it disappear	Quit

When a menu function is meaningless, its name is grayed out and you cannot select it.

### 12.2.2.11 Raising a Window

With the “Focus Follows Mouse” model, you can raise a window by simply placing the mouse over the window.

With the “Click to Focus” model, you can bring a partially concealed window to the front of the root window by clicking on the window.

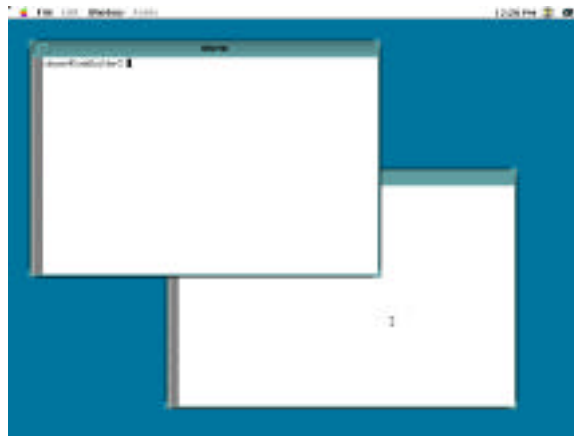
To demonstrate this with the “Click To Focus” model, create another terminal window by typing this command in the existing terminal window:

```
xterm & <CR>
```

The new window appears in the upper left corner of the display, partially covering the first window as shown in “Figure 42. A Partially Concealed Window”.

#### ***Step 1 — Position the Pointer***

Position the pointer on any visible portion of the concealed window's frame.



**Figure 42. A Partially Concealed Window**

**Step 2 — Press 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 43. The Window Revealed”.

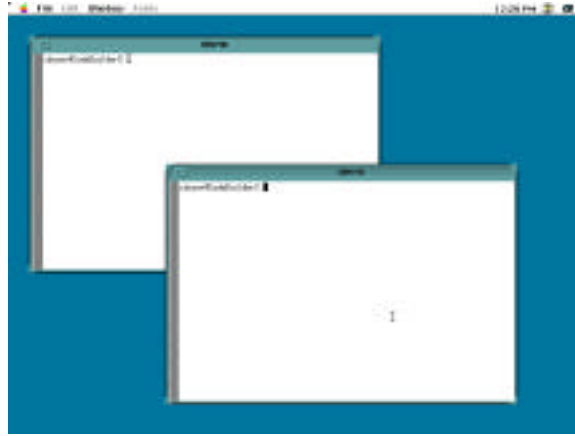


Figure 43. The Window Revealed

**12.2.2.12 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 from the Mach<sup>Ten</sup> menu bar and select “Close Window”. This removes all terminal windows from the screen (as well as the windows for clocks and similar windowed programs) and terminates the X server.

## 12.2.3 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<sup>Ten</sup> terminal window environment or other running Macintosh applications.

The X server program is the file `/usr/X11/bin/X11/XMachTen` or, by its linked name, `/usr/X11/bin/X`. By default, `xinit` will automatically run `/usr/X11/bin/X`.

You are prevented from running the X server program except by way of `xinit` under Mach<sup>Ten</sup>.

### 12.2.3.1 X Server Startup Options

By default, `xinit` will automatically run `/usr/X11/bin/X`. If you need to start the X server with options, edit the file named `.Xparams` in your home directory. This file should contain any option flags you wish to pass to the server. Startup options alter the default X server behavior and are useful for three-button mouse operation, font path redirection, and menu bar configuration, among other things.

A sample `Xparams` file is available in `/usr/X11/lib/X11/xinit/Xparams`. To make user-specific customizations, copy this file to your home directory as `.Xparams` if and only if you want to start the server with different options. If no `.Xparams` file is found, the X server uses the default options in `/usr/X11/lib/X11/xinit/Xparams`.

The following command line options for the X server may be useful on a Mach<sup>Ten</sup> system:

<code>-auth authorization-file</code>	Specifies a file which contains a collection of authorization records used to authenticate access. See also the <code>xdm</code> and <code>Xsecurity</code> manual pages.
<code>bc</code>	Disables certain kinds of error checking for bug compatibility with previous releases (e.g., to work around bugs in R2 and R3 <code>xterms</code> and toolkits).



-bs	Disables backing store support on all screens.
-co <i>filename</i>	Sets name of RGB color database.
-dpi <i>resolution</i>	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 <i>cursorFont</i>	Sets the default cursor font.
-fn <i>font</i>	Sets the default font.
-fp <i>fontPath</i>	Sets the search path for fonts. This path is a comma-separated list of directories which the X server searches for font databases.
-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.  This option may not work on PCI Macintosh systems.
-help	Prints a usage message.
-I	Causes all remaining command line arguments to be ignored.
-ld <i>kilobytes</i>	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 <i>kilobytes</i>	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.

<b>-mbm</b> <i>middle-key</i>	Changes the middle mouse button keystroke mapping. <i>&lt;middle-key&gt;</i> is entered as a decimal value. It represents the Macintosh virtual key code corresponding to the desired key. Allowable values are between 0 and 127. Refer to section “12.2.3.2 Mouse Button Mapping” for a table of keys and virtual key codes.
<b>-mbr</b> <i>right-key</i>	Changes the right mouse button keystroke mapping.
<b>-menu</b>	<p>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.</p> <p>The <i>xtmenu(n)</i> program, run from your X window manager menu environment, may be used to temporarily restore access to your Macintosh desktop. <b>NOTE:</b> This option locks out access to Finder and other running Macintosh applications!</p>
<b>-mo</b>	Specifies that the Macintosh <i>&lt;Option&gt;</i> 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” startup options, the default middle and right mouse button keystrokes become <i>&lt;Option-Left Arrow&gt;</i> and <i>&lt;Option-Right Arrow&gt;</i> , respectively. Selecting this option causes all arrow keys, including the <i>&lt;Up&gt;</i> and <i>&lt;Down&gt;</i> arrows, to function normally.
<b>-static</b>	Runs the server in “StaticColor” mode, preventing X client modification of the Macintosh color table.
<b>-su</b>	Disables “save under” support on all screens.
<b>-to</b> <i>seconds</i>	Sets the default connection time-out in seconds.
<b>-wm</b>	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.

### 12.2.3.2 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 startup options or when you configure the server to accept the <Option-mouse key> combination using the “-mo” startup 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 startup 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
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” startup 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<sup>Ten</sup> X server. Your Logitech MouseMan<sup>TM</sup> control panel would look something like this:

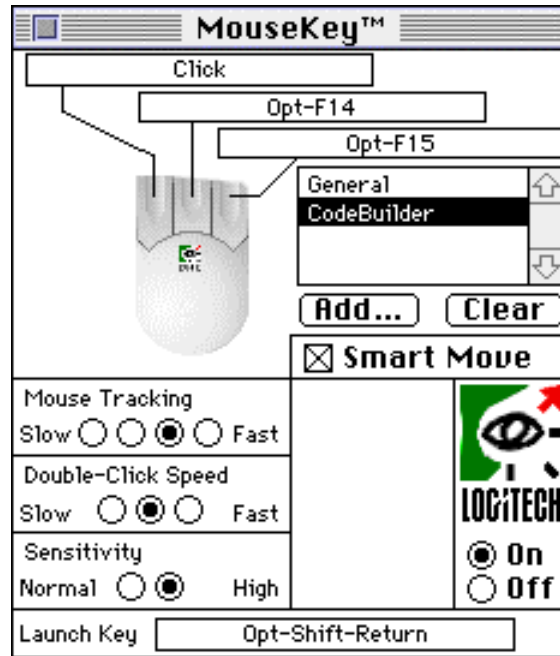


Figure 44. Logitech MouseMan<sup>TM</sup> Control Panel

The `.Xparams` configuration file in your home directory would look like this:

```
-mo -mbm 107 -mbr 113 -cmdkey meta
```

### 12.2.3.3 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 to match your country's layout when the X server is launched.

The *xmodmap* 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/X11/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:

Code	Country	Code	Country
DE	Germany	IT	Italy
DK	Denmark	NO	Norway
ES	Spain	PT	Portugal
FI	Finland	SE	Sweden
FR	France	FC	French Canadian
GB	Great Britain	CA	Canada

To change the X default U.S. keyboard layout, enter the following from a Mach<sup>Ten</sup> terminal window prior to starting the X server 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 Tenon Technical Support.

**X Meta and Alt Keys.** Under the Mach<sup>Ten</sup> X server, hot key operation (using the <Apple Command> key) normally associated with a Mach<sup>Ten</sup> 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 OpenLook Virtual 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.

### 12.2.3.4 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<sup>Ten</sup> console window.

### 12.2.3.5 X Server Performance Tuning Guide

The following tips will help you optimize the performance of the Mach<sup>Ten</sup> X server.

- 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.
- When using Finder to switch from the X desktop to Finder, a Mach<sup>Ten</sup> terminal window, or other Macintosh applications, iconify any X clients that are actively drawing to the display prior to hiding the X desktop.



### 12.2.3.6 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 are controlled by the “font path”.

Some of the directories contained in the default font path for the X server are:

/usr/X11/lib/X11/fonts/Speedo	This directory contains outline fonts for Bitstream, Inc.'s Speedo rasterizer. A single font face is provided in normal, bold, italic and bold italic.
/usr/X11/lib/X11/fonts/100dpi	This directory contains 100 dots-per-inch versions of some of the fonts in the 75dpi directory.
/usr/X11/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/X11/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.

Font databases are created by running the *mkfontdir* program in the directory containing the compiled versions of the fonts (the *.pcf* files). Whenever fonts are added to a directory, *mkfontdir* should be re-run 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.

### 12.2.3.7 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.

### 12.2.3.8 Providing Fonts Over the Network

The *xfs* program is a font server that runs on your Mach<sup>Ten</sup> system and provides fonts to other network-based X terminals and servers.

*xfs* has many configuration options which are described in detail in the online manual page. In most environments, *xfs* will run using the default configuration.

To run the font server, perform the follow steps:

- As the *root* user, start the *xfs* program:

```
/usr/bin/X11/xfs &
```

- To start the font server automatically when Mach<sup>Ten</sup> boots, add an entry to your */etc/rc.local* file:

```
if [ -x /usr/bin/X11/xfs ]; then
    ($ECHO "Starting the font server") >/dev/console
    /usr/bin/X11/xfs &
fi
```

A template */etc/rc.local* file is in */usr/local/etc/httpd/tenon/etc/rc.local*, and can be modified with a Macintosh editor.

### 12.2.4 X Display Management Under Mach<sup>Ten</sup>

The *xdm* program manages a collection of X displays, which may only be on the local host with Mach<sup>Ten</sup>. 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<sup>Ten</sup> 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<sup>Ten</sup> and the X server to be controlled by *xdm*.

### 12.2.4.1 The Mach<sup>Ten</sup> 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<sup>Ten</sup> 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<sup>Ten</sup> boot time via an entry in the */etc/rc* file. When the X server is subsequently launched with special XDMCP startup options, it communicates with the *xdm* process via the XDMCP protocol and the login window appears. XDMCP server parameters are described in section “12.2.3.1 X Server Startup Options”.

Under Mach<sup>Ten</sup>, the X server may run only after a user logs in through the Mach<sup>Ten</sup> console window and starts the *wind(8)* program. Thus, the *xdm* login window may be displayed only after entering the Mach<sup>Ten</sup> windows environment via a user account in */etc/master.passwd*.

Mach<sup>Ten</sup> contains features to automate the process of launching the operating system, logging in, and creating a default Mach<sup>Ten</sup> windows environment. 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<sup>Ten</sup> boots. This can be accomplished by the following configuration steps:

- Edit the */etc/rc.conf* file and set the `START_xdm` variable to “yes”.
- Add an account to your */etc/master.passwd* file. The account will serve as a default user shell from which actual user *xdm* sessions are started:

```
adduser xdmuser -p MyXdmPass -s /bin/csh
```

- Just prior to the invocation of the *wind* program, add the environment variable `WIND_RESET` to the *.login* file in the home directory of the *xdmuser* account:

```
setenv WIND_RESET
exec /usr/bin/wind
```

The default X desktop environment will be taken 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<sup>Ten</sup> windows environment to exit when the X server window is closed via the Mach<sup>Ten</sup> File menu.

- 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 "/usr/libexec/getty std.9600" vt100 on secure
```

Change this line to:

```
console "/usr/libexec/getty display std.9600" vt100 on secure
```

Mach<sup>Ten</sup> 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<sup>Ten</sup> windows environment.
- Follow the instructions in section “12.2.3.1 X Server Startup Options” to create custom X server parameters for the XDM managed desktop.
- Copy */usr/lib/X11/startX/startXrc* to a *.startXrc* file in the home directory of the *xdmuser* account. Edit the *.startXrc* file and change the following:
  1. Comment out the `USE_XINIT` variable.
  2. Uncomment the `XDM_METHOD` variable.
  3. Uncomment the `XDM_SERVER=<hostname>` variable and change `<hostname>` to `<localhost>`.

When the server starts up, it will communicate with *xm* to display the *xm* login window on the local display.

- You can further automate the process to display an *xm* login window when your Macintosh is powered on by using Finder to place an alias to the Mach<sup>Ten</sup> application in your *System Folder:Start Up Items* folder.

**The XDM Session.** The user session is started when a user logs into the *xm* login window. The session is controlled by an executable shell script called *.xsession* in the logged-in user's home directory. If this file is not found, *xm* runs a default session from the file */usr/lib/X11/xm/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 *xm* presents a new login window.

The *.xsession* file must have "execute" privilege. Use the command:

```
chmod 755 .xsession
```

to make the *.xsession* file executable.

## 12.2.5 X Over Serial Lines

Under Mach<sup>Ten</sup>, 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<sup>Ten</sup> is found in sections “10.15.8 Using PPP” and “10.15.9 Using SLIP”, and in <http://www.tenon.com/support/PPP>.

The `-wm` backing store X server startup 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 “12.2.3.1 X Server Startup 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.

## 12.3 Mach<sup>Ten</sup> X Window Software Overview

This release of the Mach<sup>Ten</sup> X Window software is based on the public release of X11, release 6, from MIT. Its purpose is to provide the necessary tools to allow for the development and display of X applications under Mach<sup>Ten</sup> 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 X11R6 release adapted for use under Mach<sup>Ten</sup>. Programming and server environments in the OpenLook GUI style, based on 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 <sup>Ten</sup> X Window server
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 Intrinsic 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
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 <i>Makefiles</i>
Clients	A 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 ( <i>libX*.a</i> )
/usr/X11/lib/X11	X11 configuration files, fonts and application default resource files
/usr/X11/bin	X11 executable files
/usr/include/X11	X11 <i>include</i> files for application development
/usr/X11/man/man1	X11 manual pages

### 12.3.1 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 & Sound control panel prior to running the X server. Open your Monitors & Sound control panel and select anything but the "4" or "16" Colors or Grays settings.

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 Monitors & Sound control panel settings.

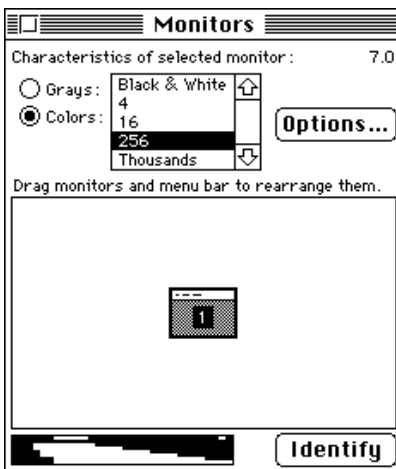


Figure 45. Monitors & Sound Control Panel

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.

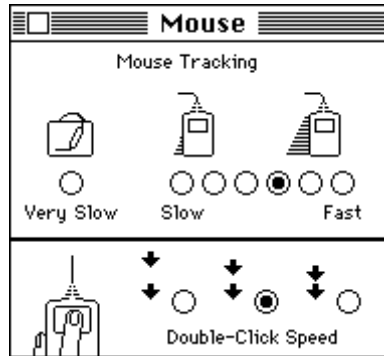


Figure 46. Mouse Control Panel

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.

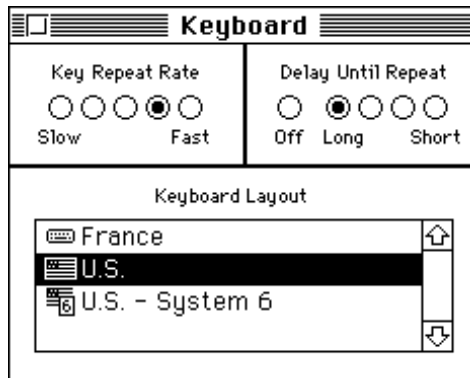


Figure 47. Keyboard Control Panel

If you have a Control Strip control panel, select the “Hide Control Strip” option when running the X server.



Figure 48. 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.

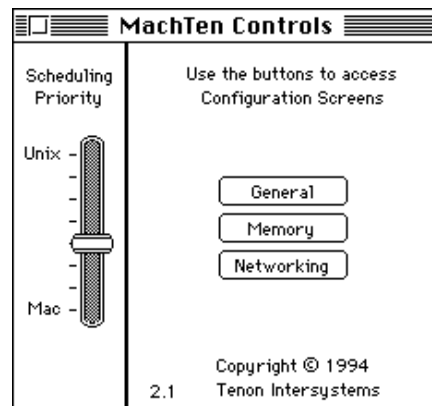


Figure 49. Mach<sup>Ten</sup> Control Panel

## 12.3.2 Getting Started With X

The 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 “12.1 The X Desktop”.
- If you want to reconfigure the default X server or learn more about its capabilities before launching it, refer to section “12.2.3 The X Server Program”. Three-button mouse, remote font server and international (non-U.S.) keyboard users must consult this section before launching the X server.
- The online manual pages *XMachTen*, *XServer* and *X* are useful and convenient references.
- General information on the operation of Mach<sup>Ten</sup> is found in this User's Guide.

## 12.3.3 Building X Applications

Most of the distributed X client executable files in the */usr/X11/bin* directory have been created from the source files in */base/src/X11/R6* (see section “2.2.2 Accessing Mach<sup>Ten</sup> Sources from the CD-ROM”). This section discusses the general procedure for creating X client applications under Mach<sup>Ten</sup>.

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* script in */usr/X11/bin/* invokes *imake* to build the *Makefile* from an *Imakefile*.

1. Be sure that the CD-ROM and *Source\_FFS* are mounted.
2. Create a working area for X11 development:

```
mkdir /base/macppc/X11
cd /base/macppc/X11
ln -s /base/src/X11
```

3. To build a single X client (e.g., *xlogo*), make the *Makefile* and executable:

```
cd /base/macppc/src/X11/R6/mit/clients/xlogo
xmkmf
make
```

4. To install the application(s) in */usr/X11/bin/*, type:

```
make install
```

5. To install the application manual page(s) in */usr/X11/man/man1*, type:

```
make install.man
```

6. 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:

7. Make the master *Makefile* using the *xmkmf* utility:

```
cd /base/macppc/X11/R6/mit/clients
xmkmf
```

8. Build the application *Makefiles* and executables:

```
make Makefiles
make
```

then follow steps 4 through 6, 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<sup>Ten</sup>.

### 12.3.3.1 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/X11/bin/xlogo &
```

### 12.3.3.2 The X11 Application Development Environment Under Mach<sup>Ten</sup>

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/X11/bin/* 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/X11/lib/X11/config/Imake.rules*. Examples of *Imakefiles* can be found under the */base/src/X11/R6/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 */usr/X11/lib/X11/config/* directory. *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<sup>Ten</sup> 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.



### 12.3.3.3 Programming Notes

Mach<sup>Ten</sup> supplies each UNIX process with a fixed 36K byte stack. 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 to 100K bytes, use the command:

```
LOCAL_LDFLAGS = -Xlstack=100000
```

Refer to section “11.0 Mach<sup>Ten</sup> Programming Environment” for further information on programming under Mach<sup>Ten</sup>.



---

# APPENDIX A

---

## GNAT For The Macintosh

### What is GNAT?

GNAT is a high-quality compiler for the entire Ada-95 language, including all annexes. The compiler is integrated with the GCC compiler system. The entire GNAT product is distributed freely, with sources, under the Copyleft policy of the Free Software Foundation. GNAT is part of the GNU project.

### What is GNAT-Mac?

GNAT-Mac is a GNAT compiler hosted on the Macintosh that produces code for that platform. The compiler is included with Mach<sup>Ten</sup>. It also runs under Power Mach<sup>Ten</sup> and Professional Mach<sup>Ten</sup>, UNIX systems that run as Macintosh applications. The compiler runs as a UNIX program.

### The GNAT-Mac Development Project

GNAT for the Macintosh is the product of a team effort. This project has been supported in part by the Ada Technology Insertion Program – Partners (ATIP-P), under contract C96-175929. Support was also provided through INEL, Idaho National Engineering Laboratory. The GNAT-Mac Technical Team consisted of:

- Professor Michael Feldman
- Jim Hopper
- Dr. John Matthews
- Dr. Art Evans

This team worked with Ada Core Technologies and Tenon to develop GNAT-Mac and the Mach<sup>Ten</sup> product.

Ada Core Technologies provides commercial support for GNAT and related software, including maintenance, porting of Ada83 applications to Ada95, porting of GNAT to new hardware platforms, training, and consulting. ACT can be contacted at:

Ada Core Technologies  
73 Fifth Avenue, Suite 11-B  
New York City, NY 10003  
info@gnat.com

ACT maintains a site that holds the latest versions of the Macintosh compilers and all documentation at:

`ftp://pubmcada:@gnat-mac.com/usr/users/macada/public`

or log in to: `ftp://gnat-mac.com` as user `pubmcada`, no password, and `cd` to `/usr/users/macada/public`.

ACT provides GNAT compilers for all platforms (including the Mac) at this site:

`ftp://cs.nyu.edu/pub/gnat`

As of this writing, the relevant files are in directories `mac68k` and `powermac`.

For other mirror sites, see the link "FTP and Mirror Sites" on the ACT Home Page, <http://www.gnat.com>.

## What is Ada?

Ada is the language of choice where software engineering, reliability, cost-effectiveness, large-scale development, and reuse matter. It is an industrial-strength programming language for real-time systems and DSP programming. Ada is currently used for applications in such domains as financial services, avionics, aeronautics, air traffic control, telecommunications, medical devices, power plants, railroads, astrophysics, satellites and defense, to name a few.

The Internet provides extensive resources for learning about all aspects of Ada. We recommend the Ada Home Page (<http://www.adahome.com>) for up-to-date links, directly or indirectly, to every Ada-related Ada resource on the Internet. For beginners, there are links to Ada tutorial material; for more advanced users there are links to the documentation defining the language. To learn more about HBAP, try the "Welcome Tour" link on the HBAP Home Page.

---

## Documentation Supplement for GNAT

---

### Part 1: An Introduction to Some GNAT Tools

Michael B. Feldman, The George Washington University  
Copyright 1996, Michael B. Feldman. All Rights Reserved.  
last revised September 1996

The document serves as an introduction to some of the tools provided with the GNAT (GNU Ada 95) compilation system. This document does not replace *gnatinfo.txt*, the “official” GNAT user manual, but rather selectively augments it with examples of tool use and some helpful hints. All these tools have many more options and flags than we shall cover here; please refer to *gnatinfo.txt* for further details.

#### A1.1 Using *gnatmake*

*gnatmake* is the GNAT tool that builds an executable program from a set of source files, typically an Ada main program and a set of library units like packages. *gnatmake* has many options, including compiler, binder, and linker options to be passed through to those tools, multidirectory compilation structures, and the like. Here we cover some of the most frequently-used options. Throughout, our annotations and explanations are in the typeface you are reading. Our command line input is shown like this:

```
sample command line
```

and the output is shown like this:

```
sample output line
```

Consider a program `test_factorial.adb`, a source file containing one nested subprogram, `Factorial`, which recursively computes the factorial of its positive input argument. We build an executable `test_factorial` with the command

```
gnatmake -v -g -gnat1 test_factorial
```

We have used the three options.

- v        verbose, which traces the steps in the build process
- g        include debugging info, which causes symbol table information to be included in the executable for debugging purposes. We will use this later.
- gnat1    which displays a compilation listing on the screen

Here is the screen output from this command, with our annotations:

```
GNATMAKE 3.05 (960607) Copyright 1995 Free Software
Foundation, Inc.
    "test_factorial.ali" being checked ...
    -> "test_factorial.adb" time stamp mismatch
```

`test_factorial` needs to be compiled; invoke the compiler and display the listing.

```
gcc -c -g -gnat1 test_factorial.adb
```

```
GNAT 3.05 (960607) Copyright 1991-1996 Free Software Foundation, Inc.
```

```
Compiling: test_factorial.adb (source file time stamp: 1996-09-22
15:15:47)
```

```

1. with Ada.Text_IO;
2. with Ada.Integer_Text_IO;
3. procedure Test_Factorial is
4. -----
5. --| Demonstrates the factorial function
6. --| Author: Michael B. Feldman, The George Washington University
7. --| Copyright 1996, Michael B. Feldman. All Rights Reserved.
8. --| Last Modified: August 1996
9. -----
10.
```

```
11.  Answer: Positive;
12.
13.  function Factorial (N : IN Positive) return Positive is
14.
15.  -- Computes the factorial of N (N!) recursively
16.  -- Pre : N is defined
17.  -- Post: returns N!
18.
19.      Result : Positive;
20.
21.  begin -- Factorial
22.
23.      if N = 1 then
24.          Result := 1; -- stopping case
25.      else
26.          Result := N * Factorial(N-1); -- recursion
27.      end if;
28.
29.      return Result;
30.  end Factorial;
31.
32. begin -- Test_Factorial
33.
34.  Answer := Factorial(4);
35.  Ada.Text_IO.Put(Item => "The value of 4! is ");
36.  Ada.Integer_Text_IO.Put(Item => Answer, Width => 11);
37.  Ada.Text_IO.New_Line;
38.
39. end Test_Factorial;
```

39 lines: No errors

Successful compilation; invoke the binder and linker.

```
gnatbind -x test_factorial.ali
gnatlink -g test_factorial.ali
```

Now we run the executable:

```
./test_factorial
```

The value of 4! is 24

## A1.2 Overriding the GNAT default file-naming conventions

GNAT **requires** that each source file contain only one compilation unit (see *gnatchop* in the GNAT User's Guide for details on how to split multi-unit files). GNAT *prefers* that each file name agree with the name of the unit in that file, e.g., *Test\_Factorial* is in the file *test\_factorial.adb*. Note that the file name is always in lower-case, and that the file extension should be *.ads* for a package spec and *.adb* otherwise.

These conventions are sometimes inconvenient, for example, when porting in Ada source code from another compiler with different conventions. It is possible to override these preferences, using a configuration file called *gnat.adc*. If this file is present in the current directory, *gnatmake* will use it to map the file names to the unit names. Let's look at a set of files making up a program to do rational-number arithmetic. Here are the contents of *gnat.adc*.

```
pragma Source_File_Name
  (Unit_Name => Rationals, Spec_File_Name => "prog21.ads");
pragma Source_File_Name
  (Unit_Name => Rationals, Body_File_Name => "prog22.adb");
pragma Source_File_Name
  (Unit_Name => Rationals.IO, Spec_File_Name =>
   "prog23.ads");
pragma Source_File_Name
  (Unit_Name => Rationals.IO, Body_File_Name =>
   "prog24.adb");
pragma Source_File_Name
  (Unit_Name => Test_Rationals_1, Body_File_Name =>
   "prog25.adb");
```

*Rationals* provide a simple rational number package; *Rationals.IO* provides *Get* and *Put* operations for rationals, and *Test\_Rationals\_1* is a simple demonstration of the package. To get a listing of the rationals spec, we type:

```
gnatmake -gnats -gnat1 prog21.ads
```

(the *-gnats* option does syntax checking only)

```
gcc -c -gnats -gnat1 prog21.ads
```



GNAT 3.05 (960607) Copyright 1991-1996 Free Software Foundation, Inc.

Checking: prog21.ads (source file time stamp: 1996-09-22 14:14:00)

```
1. package Rationals is
2. -----
3. --|
4. --| Specification of the abstract data type for representing
5. --| and manipulating rational numbers.
6. --| All rational quantities in this package are initialized
7. --| to 0/1.
8. --|
9. --| Author: Michael B. Feldman, The George Washington University
10. --| Copyright 1996, Michael B. Feldman. All Rights Reserved.
11. --| Last Modified: August 1996.
12. --|
13. -----
14.
15.   type Rational is private;
16.
17.   ZeroDenominator: exception;
18.
19.   function "/" (X : Integer; Y : Integer) return Rational;
20.   -- constructor:
21.   -- Pre :    X and Y are defined
22.   -- Post:    returns a rational number
23.   --   If Y > 0, returns Reduce(X,Y)
24.   --   If Y < 0, returns Reduce(-X,-Y)
25.   -- Raises: ZeroDenominator if Y = 0
26.
27.   function "+"(R1 : Rational; R2 : Rational) return Rational;
28.   -- dyadic arithmetic constructor:
29.   -- Pre : R1 and R2 are defined
30.   -- Post: returns the rational sum of R1 and R2
31.
32. private
33. -- A record of type Rational consists of a pair of Integer values
34. -- such that the first number represents the numerator of a
   rational
35. -- number and the second number represents the denominator.
36.
37.   type Rational is record
38.     Numerator : Integer := 0;
39.     Denominator: Positive := 1;
40.   end record;
```

```
41. end Rationals;
```

```
41 lines: No errors
```

We do the same for the spec of the rational input/output package:

```
gnatmake -gnats -gnatl prog23.ads
```

```
gcc -c -gnats -gnatl prog23.ads
```

```
GNAT 3.05 (960607) Copyright 1991-1996 Free Software Foundation, Inc.
```

```
Checking: prog23.ads (source file time stamp: 1996-09-22 14:14:00)
```

```
1. with Ada.Text_IO;
2. package Rationals.IO is
3. -----
4. --| Specification of the input/output child package for Rationals
5. --|
6. --| Author: Michael B. Feldman, The George Washington University
7. --| Copyright 1996, Michael B. Feldman. All Rights Reserved.
8. -----
9.
10.  procedure Get (Item : out Rational);
11.  -- Pre : None
12.  -- Post: The first integer number read is the numerator of Item;
13.  --       the second integer number is the denominator of Item.
14.  --       A "/" between the two numbers is optional.
15.  --       The Rational constructor "/" is called
16.  --       to produce a rational in reduced form.
17.
18.  procedure Put (Item : in Rational);
19.  -- Pre : Item is defined
20.  -- Post: displays the numerator and denominator of Item.
21.
22. end Rationals.IO;
```

```
22 lines: No errors
```

Now we are ready to build `Test_Rationals_1`:

```
gnatmake -g -v -gnatl -f prog25.adb
```

```
GNATMAKE 3.05 (960607) Copyright 1995 Free Software Foundation, Inc.  
gcc -c -g -gnatl prog25.adb
```

```
GNAT 3.05 (960607) Copyright 1991-1996 Free Software Foundation, Inc.
```

```
Compiling: prog25.adb (source file time stamp: 1996-09-22 14:14:01)
```

```
1. with Ada.Text_IO;  
2. with Rationals; use type Rationals.Rational;  
3. with Rationals.IO;  
4. procedure Test_Rationals_1 is  
5. -----  
6. --| Very rudimentary test of package Rationals and Rationals.IO  
7. --|  
8. --| Author: Michael B. Feldman, The George Washington University  
9. --| Copyright 1996, Michael B. Feldman. All Rights Reserved.  
10. --| Last Modified: July 1995  
11. -----  
12.  
13.   A: Rationals.Rational;  
14.   B: Rationals.Rational;  
15.   C: Rationals.Rational;  
16.   D: Rationals.Rational;  
17.   E: Rationals.Rational;  
18.   F: Rationals.Rational;  
19.  
20. begin -- Test_Rationals_1  
21.  
22.   A := 1/3;  
23.   B := 2/(-4);  
24.   Ada.Text_IO.Put(Item => "A = ");  
25.   Rationals.IO.Put(Item => A);  
26.   Ada.Text_IO.New_Line;  
27.   Ada.Text_IO.Put(Item => "B = ");  
28.   Rationals.IO.Put(Item => B);  
29.   Ada.Text_IO.New_Line;  
30.  
31.   -- Read in rational numbers C and D.  
32.   Ada.Text_IO.Put(Item => "Enter rational number C > ");  
33.   Rationals.IO.Get(Item => C);
```

```

34. Ada.Text_IO.Put(Item => "Enter rational number D > ");
35. Rationals.IO.Get(Item => D);
36. Ada.Text_IO.New_Line;
37.
38. E := A + B;
39. Ada.Text_IO.Put(Item => "E = A + B is ");
40. Rationals.IO.Put(Item => E);
41. Ada.Text_IO.New_Line;
42.
43. F := C + D;
44. Ada.Text_IO.Put(Item => "F = C + D is ");
45. Rationals.IO.Put(Item => F);
46. Ada.Text_IO.New_Line;
47.
48. Ada.Text_IO.Put(Item => "A + E + F is ");
49. Rationals.IO.Put(Item => A + E + F);
50. Ada.Text_IO.New_Line;
51.
52. end Test_Rationals_1;

```

52 lines: No errors

```
gcc -c -g -gnat1 prog22.adb
```

*gnatmake* has discovered that *Rationals* must be compiled (no *.ali* or *.o* file was present). Note that *gnatmake* goes directly to the *rationals* body; GNAT compiles its spec “on the fly” and produces no listing of the spec. That is why we did the earlier syntax-check steps, just to get the listings.

GNAT 3.05 (960607) Copyright 1991-1996 Free Software Foundation, Inc.

Compiling: prog22.adb (source file time stamp: 1996-09-22 14:14:00)

```

1. package body Rationals is
2.
3. -----
4. --| Body of the abstract data type for representing
5. --| and manipulating rational numbers.
6. --|
7. --| Author: Michael B. Feldman, The George Washington University
8. --| Copyright 1996, Michael B. Feldman
9. --| Last Modified: August 1996
10. -----
11.

```

```
12.  -- local function GCD, not provided to clients
13.
14.  function GCD(M: Positive; N: Positive) return Positive is
15.  -- finds the greatest common divisor of M and N
16.  -- Pre: M and N are defined
17.  -- Post: returns the GCD of M and N, by Euclid's Algorithm
18.
19.      R      : Natural;
20.      TempM: Positive;
21.      TempN: Positive;
22.
23.  begin -- GCD
24.
25.      TempM := M;
26.      TempN := N;
27.
28.      R := TempM rem TempN;
29.
30.      while R /= 0 loop
31.          TempM := TempN;
32.          TempN := R;
33.          R := TempM rem TempN;
34.      end loop;
35.
36.      return TempN;
37.
38.  end GCD;
39.
40.  -- exported operations
41.
42.  function "/" (X : Integer; Y : Integer) return Rational is
43.      G: Positive;
44.  begin -- "/"
45.
46.      if Y = 0 then
47.          raise ZeroDenominator;
48.      end if;
49.
50.      if X = 0 then
51.          return (Numerator => 0, Denominator => 1);
52.      end if;
53.
54.      G := GCD(abs X, abs Y);
55.      if Y > 0 then
56.          return (Numerator => X/G, Denominator => Y/G);
```

```

57.     else
58.         return (Numerator => (-X)/G, Denominator => (-Y)/G);
59.     end if;
60.
61. end "/";
62.
63. -- dyadic arithmetic operator
64.
65. function "+"(R1 : Rational; R2 : Rational) return Rational is
66.     N: Integer;
67.     D: Positive;
68. begin -- "+"
69.     N := R1.Numerator * R2.Denominator + R2.Numerator *
70.         R1.Denominator;
71.     D := R1.Denominator * R2.Denominator;
72.     return N/D; -- compiler will use Rational constructor here!
73. end "+";
74. end Rationals;

```

74 lines: No errors

Similarly, *gnatmake* invokes a compilation for *Rationals.IO*.

```
gcc -c -g -gnat1 prog24.adb
```

GNAT 3.05 (960607) Copyright 1991-1996 Free Software Foundation, Inc.

Compiling: prog24.adb (source file time stamp: 1996-09-22 14:14:00)

```

1. with Ada.Text_IO;
2. with Ada.Integer_Text_IO;
3. package body Rationals.IO is
4. -----
5. --| Body of the input/output child package for Rationals
6. --|
7. --| Author: Michael B. Feldman, The George Washington University
8. --| Copyright 1996, Michael B. Feldman. All Rights Reserved
9. --| Last Modified: August 1996
10. -----
11.
12.     -- input procedure
13.
14.     procedure Get (Item : out Rational) is

```

```
15.
16.     N: Integer;
17.     D: Integer;
18.     Dummy: Character; -- dummy character to hold the "/"
19.
20.     begin -- Get
21.
22.         Ada.Integer_Text_IO.Get(Item => N);
23.         Ada.Text_IO.Get (Item => Dummy);
24.         Ada.Integer_Text_IO.Get(Item => D);
25.         Item := N/D;
26.
27.     end Get;
28.
29.     -- output procedure
30.
31.     procedure Put (Item : in Rational) is
32.
33.     begin -- Put
34.
35.         Ada.Integer_Text_IO.Put(Item => Item.Numerator, Width => 1);
36.         Ada.Text_IO.Put(Item => '/');
37.         Ada.Integer_Text_IO.Put(Item => Item.Denominator, Width => 1);
38.
39.     end Put;
40.
41. end Rationals.IO;
```

41 lines: No errors

Finally, *gnatmake* invokes the binder and linker.

```
gnatbind -x prog25.ali
gnatlink -g prog25.ali
```

Now we execute the demonstration program several times:

```
./prog25
```

```
A = 1/3
B = -1/2
Enter rational number C > 2/4
Enter rational number D > 9/6
```

```
E = A + B is -1/6
F = C + D is 2/1
A + E + F is 13/6
```

```
./prog25
```

```
A = 1/3
B = -1/2
Enter rational number C > 0/1
Enter rational number D > 9/6
```

```
E = A + B is -1/6
F = C + D is 3/2
A + E + F is 5/3
```

```
./prog25
```

```
A = 1/3
B = -1/2
Enter rational number C > 1/0
```

```
raised RATIONALS.ZERODENOMINATOR
```

Here we entered a zero for the denominator of a rational. This is not allowed mathematically, so the exception is raised (somewhere in the rationals package) and propagated out of the main program.

We can get a full traceback showing where the exception was raised and how it propagated, but we need *gdb* for that.



### A1.3 Using *gdb* to get a simple traceback

GNAT does not come with its own debugger, or even a built-in traceback facility. Instead, it depends upon the GNU debugger, *gdb*. Let's use *gdb* to get a traceback from our rationals demonstration. We type:

```
gdb prog25
```

and we are placed in *gdb*'s command processor. *gdb*'s prompts are indicated by (*gdb*).

```
GDB is free software and you are welcome to distribute copies
of it under certain conditions; type "show copying" to see
the conditions. There is absolutely no warranty for GDB; type
"show warranty" for details.
```

```
GDB 4.15.1.gnat.1.10
```

```
Copyright 1995 Free Software Foundation, Inc.
```

```
(gdb) break __gnat_unhandled_exception
```

We have indicated that we wish *gdb* to stop the program and accept more commands, if an unhandled exception is raised. The address of the breakpoint is given; this will, in general, vary from machine to machine and even execution to execution. The file *a-raise.c* is part of the GNAT runtime and not of concern to us here.

```
Breakpoint 1 at 0x17dec: file a-raise.c, line 65
```

Now we tell *gdb* to run the program. It will run normally, unless that breakpoint is reached.

```
(gdb) run
```

```
Starting program: prog25
```

```
Breakpoint 1 at 0x1fa328c: file a-raise.c, line 65.
```

```
A = 1/3
```

```
B = -1/2
```

```
Enter rational number C > 1 0
```

```
Breakpoint 1, __gnat_unhandled_exception (except=0x36abb58)
at a-raise.c:65
```

a-raise.c:65: No such file or directory.

Here we stop at the breakpoint, because we entered that zero denominator. Ignore the last line about a-raise.c. We request a traceback (which *gdb* calls a *backtrace*):

(gdb) **backtrace**

```
#0  __gnat_unhandled_exception (except=0x36abb58) at a-raise.c:65
#1  0x1fa33c4 in __gnat_raise_nodefer_with_msg
    (except=0x36abb58)
    at a-raise.c:108
#2  0x1fa3438 in __gnat_raise_nodefer (except=0x36abb58) at
a-raise.c:120
#3  0x1fa34cc in __gnat_raise (except=0x36abb58) at a-
raise.c:130
#4  0x1fa2510 in rationals. "/" (x=1, y=0) at prog22.adb:47
#5  0x1fa288c in rationals.io.get (item={numerator = 0,
denominator = 1})
    at prog24.adb:25
#6  0x1fa2b08 in test_rationals_1 () at prog25.adb:33
#7  0x1f8b64c in main (argc=1, argv=0x1f8a3ec,
envp=0x1f8a3f4) at b_prog25.c:40
#8  0x1f8b4e8 in __start (=0x4186001c, =0x81620008, =0x1)
#9  0x1f8b4a0 in __start (=0x1f8a59f, =0x1f8a5c6, =0x1f8a5d1)
```

Refer back to the listings above. The interesting part of the traceback, lines 4-6, shows that the exception was raised in the function "/" at line 22 of *Rationals*, which was called from the procedure *Get* at line 25 of *Rationals.IO*, which in turn was called from line 33 of *Test\_Rationals\_1*. The first four traceback lines, and the last 3 lines, give trace information from the GNAT runtime system. Normally you can ignore these lines.

(gdb) **quit**

The program is running. Quit anyway (and kill it)? (y or n) **y**

## A1.4 Tracing a Program with *gdb*

We can show some more *gdb* commands by returning to the program *test\_factorial*. Here we will break not on an exception, but every time the *Factorial* function is called. Since *Factorial* is recursive, we will stop at every new level of recursion.

```
gdb test_factorial
```

GDB is free software and you are welcome to distribute copies of it under certain conditions; type "show copying" to see the conditions. There is absolutely no warranty for GDB; type "show warranty" for details. GDB 4.15.1.gnat.1.10, Copyright 1995 Free Software Foundation, Inc.

First we stop at the first executable statement of the main program:

```
(gdb) break test_factorial  
Breakpoint 1 at 0x16ee4: file test_factorial.adb, line 34.
```

Now we set a break for each time we call *Factorial*:

```
(gdb) break factorial  
Breakpoint 2 at 0x16dd0: file test_factorial.adb, line 23.
```

```
(gdb) run  
Starting program: test_factorial  
Breakpoint 1 at 0x2f6ef24: file test_factorial.adb, line 34.  
Breakpoint 2 at 0x2f6ee10: file test_factorial.adb, line 23.
```

```
Breakpoint 1, test_factorial () at test_factorial.adb:34  
34      Answer := Factorial(4);
```

and we continue the run:

```
(gdb) continue  
Continuing.
```

```
Breakpoint 2, test_factorial.factorial (n=4) at  
test_factorial.adb:23  
23      if N = 1 then
```

Now we will set a watchpoint on the variable *Result*, so that the program will stop every time *Result* acquires a new value.

```
(gdb) watch result
Watchpoint 3: test_factorial.factorial::result
```

```
(gdb) continue
Continuing.
Watchpoint 3: test_factorial.factorial::result
```

```
Old value = 57185676
New value = 57179476
0x1efec48 in test_factorial.factorial (n=32402616) at
test_factorial.adb:13
13      function Factorial (N : IN Positive) return Positive
is
```

The huge values (which vary from execution to execution) suggest that *Result* is uninitialized at this point. This is true.

```
(gdb) continue
Continuing.

Breakpoint 2, test_factorial.factorial (n=3) at test_factorial.adb:23
23      if N = 1 then
```

```
(gdb) continue
Continuing.
Breakpoint 2, test_factorial.factorial (n=2) at test_factorial.adb:23
23      if N = 1 then
```

```
(gdb) continue
Continuing.
Watchpoint 3: test_factorial.factorial::result
```

```
Old value = 57179476
New value = 0
0x1efec48 in test_factorial.factorial (n=32402440) at
test_factorial.adb:13
13      function Factorial (N : IN Positive) return Positive is
```

```
(gdb) continue
Continuing.
```

```
Breakpoint 2, test_factorial.factorial (n=1) at test_factorial.adb:23
23         if N = 1 then
```

```
(gdb) continue
```

```
Continuing.
```

```
Watchpoint 3: test_factorial.factorial::result
```

```
Old value = 0
```

```
New value = 1
```

```
0x1efec64 in test_factorial.factorial (n=1) at test_factorial.adb:24
24         Result := 1;           -- stopping case
```

Now we've recursed all the way down, and start back up:

```
(gdb) continue
```

```
Continuing.
```

```
Watchpoint 3: test_factorial.factorial::result
```

```
Old value = 1
```

```
New value = 57179476
```

```
0x1efed18 in test_factorial.factorial (n=2) at test_factorial.adb:29
29         return Result;
```

```
(gdb) continue
```

```
Continuing.
```

```
Watchpoint 3: test_factorial.factorial::result
```

```
Old value = 57179476
```

```
New value = 2
```

```
test_factorial.factorial (n=2) at test_factorial.adb:29
29         return Result;
```

```
(gdb) continue
```

```
Continuing.
```

```
Watchpoint 3: test_factorial.factorial::result
```

```
Old value = 2
```

```
New value = 57179476
```

```
0x1efed18 in test_factorial.factorial (n=3) at test_factorial.adb:29
29         return Result;
```

```
(gdb) continue
```

```
Continuing.
```

```
Watchpoint 3: test_factorial.factorial::result
```

```
Old value = 57179476
New value = 6
test_factorial.factorial (n=3) at test_factorial.adb:29
29      return Result;

(gdb) continue
Continuing.
Watchpoint 3: test_factorial.factorial::result

Old value = 6
New value = 57185676
0x1efed18 in test_factorial.factorial (n=4) at test_factorial.adb:29
29      return Result;

(gdb) continue
Continuing.
Watchpoint 3: test_factorial.factorial::result

Old value = 57185676
New value = 24
test_factorial.factorial (n=4) at test_factorial.adb:29
29      return Result;

(gdb) continue
Continuing.
Watchpoint 3 deleted because the program has left the block in
which its expression is valid.
0x1efecf8 in test_factorial.factorial (n=4) at test_factorial.adb:29
29      return Result;

(gdb) continue
Continuing.
The value of 4! is          24

Program exited normally
```

Refer to the GNAT documents for a full set of documentation on *gdb*. *gdb* also provides fairly useful on-line help:

```
(gdb) help
```

```
List of classes of commands:
running -- Running the program
stack -- Examining the stack
```

```
data -- Examining data
breakpoints -- Making program stop at certain points
files -- Specifying and examining files
status -- Status inquiries
support -- Support facilities
user-defined -- User-defined commands
aliases -- Aliases of other commands
obscure -- Obscure features
internals -- Maintenance commands
```

Type "help" followed by a class name for a list of commands in that class.  
Type "help" followed by command name for full documentation.  
Command name abbreviations are allowed if unambiguous.

## A1.5 Using *gnatf* to get a cross reference

Suppose we wish to know, for a given program, all its declarations and all the declarations it uses from the packages it *WITHs*, and where (which lines) these are referenced. Looking again at our rationals demonstration, we type:

```
gnatf -x6 prog25.adb
```

*gnatf* stores the cross reference information in the file *X.ref*, so let's look at its contents.

Here we see the variables declared in *Test\_Rationals\_1*, and where in that program each is used (line and character position within line).

```
cat X.ref
```

```
%% prog25.adb          960922141401 %%
test_rationals_1 procedure 4:11
a variable 13:3
  prog25.adb {22:3 25:28 38:8 49:28}
b variable 14:3
  prog25.adb {23:3 28:28 38:12}
c variable 15:3
  prog25.adb {33:28 43:8}
d variable 16:3
  prog25.adb {35:28 43:12}
e variable 17:3
  prog25.adb {38:3 40:28 49:32}
```

```
f variable 18:3
  prog25.adb {43:3 45:28 49:36}
```

**The main program calls some procedures from *Ada.Text\_IO***

```
-- /usr/local/adainclude/a-textio.ads 960625050333 --
text_io package 30:13
  prog25.adb {1:10 24:7 26:7 27:7 29:7 32:7 34:7 36:7 39:7 41:7 44:7 46:7
48:7
  50:7}
text_io.new_line procedure 141:14
  prog25.adb {26:15 29:15 36:15 41:15 46:15 50:15}
text_io.put procedure 217:14
  prog25.adb {24:15 27:15 32:15 34:15 39:15 44:15 48:15}

-- /usr/local/adainclude/ada.ads 960625050300 --
ada package 18:9
  prog25.adb {24:3 26:3 27:3 29:3 32:3 34:3 36:3 39:3 41:3 44:3 46:3 48:3
50:3}
3}
```

**and from *Rationals*:**

```
-- prog21.ads 960922141400 --
rationals package 1:9
  prog25.adb {2:6 2:26 13:6 14:6 15:6 16:6 17:6 18:6 25:3 28:3 33:3 35:3
40:3 45
  :3 49:3}
rational private_type 15:8 37:8
  prog25.adb {2:36 13:16 14:16 15:16 16:16 17:16 18:16}
"/" function 19:12
  prog25.adb {22:9 23:9}
"+" function 27:12
  prog25.adb {38:10 43:10 49:34 49:30}
```

**and from *Rationals.IO*:**

```
-- prog23.ads 960922141400 --
io package 2:19
  prog25.adb {3:16 25:13 28:13 33:13 35:13 40:13 45:13 49:13}
io.get procedure 10:13
  prog25.adb {33:16 35:16}
io.put procedure 18:13
  prog25.adb {25:16 28:16 40:16 45:16 49:16}
```



## A1.6 Using *gnatk8*

We saw above how to override the GNAT file-naming conventions using a configuration file. But suppose we prefer to follow GNAT's preferences. We must name our files according to the units they contain. This can cause a problem if a unit's name is longer than our computer's maximum name length. For example, the Apple Macintosh has a file-name limit of 31 characters. Because our GNAT file names all end with a dot and a 3-letter extension, effectively we have a 27-character limit on the file name.

For systems with limits on file names, GNAT has an algorithm (krunch) for "crunching" its preferred file names down to the limit. On the Mac, if we add the flag *-gnatk27* to a *gnatmake* invocation, GNAT will assume that all source file names follow the krunch algorithm for 27 characters. The algorithm is a bit involved; it is documented a bit in the GNAT Users Guide. Suffice it to say that if the file name has 27 characters or less, it is used without change.

Suppose the file name is long. How do we determine its krunched name? The tool *gnatk8* gives us this capability. (This is an odd name; it comes from the fact that file names in DOS or OS/2 FAT file systems are 8 characters or less; the tool was developed originally for those systems.)

Given a unit:

```
Very_Long_Unit_Name.Long_Child_Package_Name
```

stored in a file *some\_file.adb*. We can rename this file correctly on the Mac in two steps. First, ask *gnatk8* what the name should be:

```
gnatk8 Very_Long_Unit_Name.Long_Child_Package_Name 27
```

```
verlonuninamlonchilpackname
```

Then rename the file accordingly:

```
mv some_file.adb verlonuninamlonchilpackname.adb
```



---

# APPENDIX A

## Documentation Supplement for GNAT

---

### Part 2: GNAT User's Guide

(C) Copyright 1995-1996, Ada Core Technologies, Inc.  
GNAT is free software; you can redistribute it and/or modify it under terms of the GNU General Public License as published by the Free Software Foundation; either version 2, or (at your option) any later version. GNAT is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details. You should have received a copy of the GNU General Public License along with GNAT; see file COPYING. If not, write to the Free Software Foundation, 675 Mass Ave., Cambridge, MA 02139, USA.

---

<b>Part 2: GNAT User's Guide</b>	<b>A2-1</b>
<b>A2.1.0 About This Guide</b>	<b>A2-5</b>
A2.1.1 What This Guide Contains	A2-5
A2.1.2 What You Should Know Before Reading This Guide	A2-6
A2.1.3 Related Information	A2-6
A2.1.4 Conventions	A2-6
<b>A2.2.0 Getting Started With GNAT</b>	<b>A2-7</b>
A2.2.1 Running GNAT	A2-7
A2.2.2 Running a Simple Ada Program	A2-7
A2.2.3 Running a Program With Multiple Units	A2-9
A2.2.4 Using the <i>gnatmake</i> Utility	A2-11
<b>A2.3.0 The GNAT Compilation Model</b>	<b>A2-12</b>
A2.3.1 Source Representation	A2-12
A2.3.2 Foreign Language Representation	A2-13
A2.3.2.1 Latin-1	A2-13
A2.3.2.2 Other 8-Bit Codes	A2-13
A2.3.3 File Naming Rules	A2-16
A2.3.4 Naming of GNAT Source Files	A2-17
A2.3.5 Generating Object Files	A2-18
A2.3.6 Source Dependencies	A2-19

---

A2.3.7	The Ada Library Information Files	A2-20
A2.3.8	Representation of Time Stamps	A2-23
A2.3.9	Binding an Ada Program	A2-24
A2.3.10	Mixed Language Programming	A2-25
A2.3.11	Comparison of GNAT Model With C/C++ Compilation Model	A2-25
A2.3.12	Comparison of GNAT Model With Traditional Ada Library Model	A2-26
<b>A2.4.0</b>	<b>Compiling Ada Programs With <i>gcc</i></b>	<b>A2-28</b>
A2.4.1	Compiling Programs	A2-28
A2.4.2	Switches for <i>gcc</i>	A2-29
A2.4.3	Switches for GNAT	A2-31
A2.4.3.1	Error Message Control	A2-32
A2.4.3.2	Debugging and Assertion Control	A2-36
A2.4.3.3	Runtime Checks	A2-36
A2.4.3.4	Using <i>gcc</i> for Syntax Checking	A2-37
A2.4.3.5	Using <i>gcc</i> for Semantic Checking	A2-38
A2.4.3.6	Compiling Ada 83 Programs	A2-38
A2.4.3.7	Style Checking	A2-39
A2.4.3.8	Character Set Control	A2-39
A2.4.3.9	File Naming Control	A2-40
A2.4.3.10	Subprogram Inlining Control	A2-40
A2.4.3.11	Auxiliary Output Control	A2-41
A2.4.3.12	Debugging Control	A2-41
A2.4.4	Search Paths and the Run-Time Library (RTL)	A2-43
A2.4.5	Order of Compilation Issues	A2-44
A2.4.6	Examples	A2-45

---

<b>A2.5.0</b>	<b>Binding Ada Programs With <i>gnatbind</i></b>	<b>A2-46</b>
A2.5.1	Running <i>gnatbind</i>	A2-46
A2.5.2	Consistency-Checking Modes	A2-49
A2.5.3	Error-Message Control	A2-50
A2.5.4	Output Control	A2-51
A2.5.5	Binding for Non-Ada Main Programs	A2-51
A2.5.6	Summary of Binder Switches	A2-52
A2.5.7	Command-Line Access	A2-53
A2.5.8	Search Paths for <i>gnatbind</i>	A2-53
A2.5.9	Examples of <i>gnatbind</i> Usage	A2-54
<b>A2.6.0</b>	<b>Linking Ada Programs Using <i>gnatlink</i></b>	<b>A2-56</b>
A2.6.1	Running <i>gnatlink</i>	A2-56
A2.6.2	Switches for <i>gnatlink</i>	A2-57
<b>A2.7.0</b>	<b>The GNAT Make Program <i>gnatmake</i></b>	<b>A2-58</b>
A2.7.1	Running <i>gnatmake</i>	A2-59
A2.7.2	Switches for <i>gnatmake</i>	A2-60
A2.7.3	Notes on the Command Line	A2-62
A2.7.4	How <i>gnatmake</i> Works	A2-63
A2.7.5	Examples of <i>gnatmake</i> Usage	A2-64
<b>A2.8.0</b>	<b>Handling Files With Multiple Units With <i>gnatchop</i></b>	<b>A2-65</b>
A2.8.1	Handling Files With Multiple Units	A2-65
A2.8.2	Command Line for <i>gnatchop</i>	A2-66
A2.8.3	Switches for <i>gnatchop</i>	A2-66
A2.8.4	Examples of <i>gnatchop</i> Usage	A2-67
<b>A2.9.0</b>	<b>The Front-End/Cross-Reference Utility <i>gnatf</i></b>	<b>A2-68</b>
A2.9.1	Overview of <i>gnatf</i>	A2-68
A2.9.2	Command Line of <i>gnatf</i>	A2-69
A2.9.3	Compilation Switches	A2-69
A2.9.4	Cross-Referencing Switches	A2-70
A2.9.5	Cross Reference Information and Smart Recompilation	A2-71

---

A2.9.6	File Structure	A2-72
A2.9.7	Example of <i>gnatf</i> Usage	A2-73
<b>A2.10.0</b>	<b>Filename Krunching With <i>gnatk8</i></b>	<b>A2-76</b>
A2.10.1	About <i>gnatk8</i>	A2-76
A2.10.2	Using <i>gnatk8</i>	A2-76
A2.10.3	Krunching Method	A2-77
A2.10.4	Examples of <i>gnatk8</i> Usage	A2-78
<b>A2.11.0</b>	<b>Other Utility Programs</b>	<b>A2-79</b>
A2.11.1	Using Other Utility Programs With GNAT	A2-79
A2.11.2	The Naming Scheme of GNAT	A2-79
A2.11.3	Ada Mode for Emacs	A2-80
<b>A2.12.0</b>	<b>Running and Debugging Ada Programs</b>	<b>A2-81</b>
A2.12.1	Getting Internal Debugging Information	A2-81
A2.12.2	GNAT Crashes	A2-81
A2.12.3	Using <i>gdb</i>	A2-83
<b>A2.13.0</b>	<b>Performance Considerations</b>	<b>A2-84</b>
A2.13.1	Controlling Runtime Checks	A2-84
A2.13.2	Optimization Levels	A2-85
A2.13.3	Inlining of Subprograms	A2-86

## A2.1.0 About This Guide

This guide describes the language itself, as well as the various utilities that allow you to manipulate GNAT code.

### A2.1.1 What This Guide Contains

This guide contains the following chapters:

- **Getting Started With GNAT.** Describes how to get started running and compiling with GNAT, the Ada programming environment.
- **The GNAT Compilation Model.** Describes the compilation model used by GNAT.
- **Compiling Ada Programs With *gcc*.** Describes how to compile Ada programs with *gcc*, the Ada compiler.
- **Binding Ada Programs With *gnatbind*.** Describes how to perform binding in Ada programs with *gnatbind*, the GNAT binding utility.
- **Linking Ada Programs Using *gnatlink*.** Describes *gnatlink*, a program that provides for linking using the GNAT runtime library to construct a program consisting of a mix of Ada and C sources.
- **The GNAT Make Program *gnatmake*.** Describes *gnatmake*, a utility that automatically determines and compiles the set of sources needed by an Ada compilation unit.
- **Handling Files With Multiple Units With *gnatchop*.** Describes *gnatchop*, a utility that allows you to preprocess a file and split it into several other files, one for each compilation unit.
- **The Front-end/Cross-Reference Utility *gnatf*.** Discusses *gnatf*, a modified version of the GNAT compiler.
- **Filename Krunching With *gnatk8*.** Describes the *gnatk8* filename krunching utility.
- **Other Utility Programs.** Discusses several other GNAT utilities, including *gnatk8* and *gnatf*. This chapter also contains information on cross references, smart recompilation, and cross-reference file structure.
- **Running and Debugging Ada Programs.** Describes how to run and debug Ada programs.
- **Performance Considerations.** Reviews the trade offs between using defaults or options in program development.

## A2.1.2 What You Should Know Before Reading This Guide

This user's guide assumes that you are familiar with Ada 95 language, as described in the International Standard ANSI/ISO/IEC-8652:1995, Jan 1995.

## A2.1.3 Related Information

For further information about related tools, refer to the following documents:

- GNAT Reference Manual, which contains all reference material for the Silicon Graphics Ada programming language.
- Ada 95 Language Reference Manual, which contains all reference material for the Ada 95 programming language.

## A2.1.4 Conventions

Following are examples of the typographical and graphic conventions used in this guide:

- Functions, utility program names, standard names, **and** classes.
- 'Option flags'
- '*Filenames*,' 'button names', **and** 'field names'.
- Variables.
- *Emphasis*.
- [optional information or parameters]
- Examples are described by text and  
then shown this way.



## A2.2.0 Getting Started With GNAT

This chapter describes the usual ways of using GNAT to compile Ada programs.

### A2.2.1 Running GNAT

Three steps are needed to create an executable file from an Ada source file:

1. The source file must first be compiled.
2. The source file then must be bound using the GNAT binder.
3. All appropriate object files must be linked to produce an executable.

### A2.2.2 Running a Simple Ada Program

Any editor may be used to prepare an Ada program. If emacs is used, the optional Ada mode may be helpful in laying out the program. The program text is a normal text file. We will suppose in our initial example that you have used your editor to prepare the following file:

```
with Text_IO; use Text_IO;
procedure Hello is
begin
  Put_Line ("Hello WORLD!");
end Hello;
```

This file should be named *'hello.adb'*. GNAT requires each file contain a single unit whose file name corresponds to the unit name with periods replaced by hyphens and whose extension is *'ads'* for a spec and *'adb'* for a body.

You can compile the program using one of the following commands:

```
gcc -c hello.adb
gnatmake -c hello
gnatmake -c hello.adb
```

*gcc* is the command used to access the compiler. This compiler is capable of compiling programs in several languages including Ada 95 and C. It determines you have given it an Ada program by the extension (‘.ads’ or ‘.adb’), and will call the GNAT compiler to compile the specified file.

The ‘-c’ switch is required. It tells *gcc* to do a compilation. (For C programs, *gcc* can also do linking, but this capability is not used directly for Ada programs, so the ‘-c’ switch must always be present.)

This compile command generates a file ‘*hello.o*’ which is the object file corresponding to your Ada program. It also generates a file ‘*hello.ali*’ which contains additional information used to check that an Ada program is consistent. To get an executable file, either use *gnatmake* or use *gnatbind* followed by *gnatlink*.

*gnatmake* is a master program which invokes all of the required *gcc* tools in the correct order and compiles all necessary Ada units when requested to do so.

```
gnatmake hello
```

or

```
gnatbind -x hello.ali
gnatlink -o hello hello.ali
```

The result is an executable program called ‘*hello*’, which can be run using the normal Unix command

```
./hello
```

and, if all has gone well, you will see

```
Hello WORLD!
```

appear in response to this command.

## A2.2.3 Running a Program With Multiple Units

Consider a slightly more complicated example that has three files, a main program, and the spec and body of a package:

```
package Greetings is
  procedure Hello;
  procedure Goodbye;
end Greetings;

with Text_IO; use Text_IO;
package body Greetings is
  procedure Hello is
  begin
    Put_Line ("Hello WORLD!");
  end Hello;
  procedure Goodbye is
  begin
    Put_Line ("Goodbye WORLD!");
  end Goodbye;
end Greetings;

with Greetings;
procedure Gmain is
begin
  Greetings.Hello;
  Greetings.Goodbye;
end Gmain;
```

Following the one-unit-per-file rule, prepare this program in the following three separate files:

```
'greetings.ads'
  spec of package Greetings
'greetings.adb'
  body of package Greetings
'gmain.adb'
  body of main program
```

Compile the program in steps: one for the package, and one for the main program. Unlike the case in some other Ada compilers, there is no required order of compilation and, in particular, it is fine to compile the main program first:

```
gcc -c gmain.adb
gcc -c greetings.adb
```

Notice you do not need to compile *'greetings.ads'*. GNAT does not require that you compile library specs or library generic packages. If you want, you can submit these units to the compiler to be checked for correctness, using the *'-gnatc'* switch:

```
gcc -gnatc -c greetings.ads
```

Once all the necessary units have been compiled, you bind and link them as previously, using *gnatbind* and *gnatlink* as follows:

```
gnatbind gmain.ali
gnatlink -o gmain gmain.ali
```

A better approach is to simply use *gnatmake* as described in the following section.

## A2.2.4 Using the *gnatmake* Utility

As you work on a program, you keep track of which units you modify and make sure you not only recompile these units, but also any units that depend on units you have modified. For example, in the preceding case, if you edit *gmain.adb*, you only need recompile that file. But if you edit *greetings.ads*, you must recompile both *greetings.adb* and *gmain.adb*, since both files contain units that depend on *greetings.ads*.

*gnatbind* will warn you if you forget one of these compilation steps, so it is never possible to generate an inconsistent program as a result of forgetting to do a compilation, but it can be annoying to keep track of the dependencies. One approach would be to use a *Makefile*, but the trouble with make files is that the dependencies may change as you change the program, and you must make sure that the *Makefile* is kept up to date.

The *gnatmake* utility takes care of these details automatically. Invoke it as follows:

```
gnatmake gmain.adb
```

The argument is the file containing the main program. *gnatmake* examines the environment, automatically recompiles any files that need recompiling, and binds and links the resulting set of object files, generating the executable file, *gmain*.

## A2.3.0 The GNAT Compilation Model

This chapter describes the compilation model used by GNAT. Although similar to that used by other languages, such as C and C++, this model is substantially different from the traditional Ada compilation models, which are based on a library. The model is initially described without reference to this traditional model. If you have not previously used an Ada compiler, you need only read the first part of this chapter. The last section describes and discusses the differences between the GNAT model and the traditional Ada compiler models. If you have used other Ada compilers, you may find this section helps you to understand those differences.

### A2.3.1 Source Representation

Ada source programs are represented in standard text files, using Latin-1 coding. Latin-1 is ASCII with the additional characters used for representing foreign languages (see section “3.2 Foreign Language Representation” for support of non-USA character sets). The format effector characters are represented using their standard ASCII encodings, as follows:

VT	Vertical tab	16#0B#
HT	Horizontal tab	16#09#
CR	Carriage return	16#0D#
LF	Line feed	16#0A#
FF	Form feed	16#0C#

The end of physical lines is marked by any of the following sequences: LF, CR, CR-LF, or LF-CR. Standard Unix files simply use LF to terminate physical lines. The other combinations are recognized to provide convenient processing for files imported from other operating systems. For example, files imported from MS-DOS on a PC are likely to have lines ended by CR-LF.

The end of a source file is normally represented by the physical end of file. However the control character 16#1A# (SUB) is also represented as signalling the end of the source file. Again, this is provided for compatibility with imported MS-DOS files where this control code is used to represent the end of file.

Each file contains a single Ada compilation unit, including any pragmas associated with the unit. For example, this means you must place a package declaration (a package **spec**) and the corresponding body in separate files. An Ada **compilation** (which is a sequence of compilation units) is represented using a sequence of files. Similarly, you will place each subunit or child unit in a separate file.

## A2.3.2 Foreign Language Representation

GNAT supports the standard character sets defined in Ada 95:

### A2.3.2.1 Latin-1

The basic character set is Latin-1. This character set is defined by ISO standard 8859, part 1. The lower half (character codes 16#00# . . . 16#7F#) is identical to standard ASCII coding, but the upper half is used to represent additional characters. This includes extended letters used by European languages, such as the umlaut used in German.

For a complete list of Latin-1 codes and their encodings, see the source file of library unit `Ada.Characters.Latin_1` in file `'a-chlat1.ads'`.

You may use any of these extended characters freely in character or string literals. In addition, the extended characters that represent letters can be used in identifiers.

### A2.3.2.2 Other 8-Bit Codes

GNAT also supports several other 8-bit coding schemes:

- Latin-2** Latin-2 letters allowed in identifiers, with uppercase and lowercase equivalence.
- Latin-3** Latin-3 letters allowed in identifiers, with uppercase and lower case equivalence.
- Latin-4** Latin-4 letters allowed in identifiers, with uppercase and lower case equivalence.

**IBM PC (code page 437)**

This code page is the normal default for PCs in the U.S. It corresponds to the original IBM PC character set. This set has some, but not all, of the extended Latin-1 letters, but these letters do not have the same encoding as Latin-1. In this mode, these letters are allowed in identifiers with uppercase and lowercase equivalence.

**IBM PC (code page 850)**

This code page is a modification of 437 extended to include all the Latin-1 letters, but still not with the usual Latin-1 encoding. In this mode, all these letters are allowed in identifiers with uppercase and lower case equivalence.

**Full Upper 8-bit**

Any character in the range 80-FF allowed in identifiers, and all are considered distinct. In other words, there are no uppercase and lower case equivalences in this range.

**No Upper-Half**

No upper-half characters in the range 80-FF are allowed in identifiers. This gives Ada 95 compatibility for identifier names. For precise data on the encodings permitted, and the uppercase and lower case equivalences that are recognized, see the file 'csets.adb' in the GNAT compiler sources.

**Wide Character Coding**

GNAT allows wide character codes to appear in character and string literals, and also optionally in identifiers, using the following possible encoding schemes:

**Hex Coding**

In this encoding, a wide character is represented by the following five character sequence:

```
ESC a b c d
```

Where *a*, *b*, *c*, *d* are the four hexadecimal characters (using uppercase letters) of the wide character code. For example, ESC A345 is used to represent the wide character with code 16#A345#. This scheme is compatible with use of the full Wide\_Character set.



**Upper-Half Coding**

The wide character with encoding 16#abcd# where the upper bit is on (in other words, "a" is in the range 8-F) is represented as two bytes, 16#ab# and 16#cd#. The second byte may never be a format control character, but is not required to be in the upper half. This method can be also used for shift-JIS or EUC, where the internal coding matches the external coding.

**Shift JIS Coding**

A wide character is represented by a two-character sequence, 16#ab# and 16#cd#, with the restrictions described for upper-half encoding as described above. The internal character code is the corresponding JIS character according to the standard algorithm for Shift-JIS conversion. Only characters defined in the JIS code set table can be used with this encoding method.

**EUC Coding**

A wide character is represented by a two-character sequence 16#ab# and 16#cd#, with both characters being in the upper half. The internal character code is the corresponding JIS character according to the EUC encoding algorithm. Only characters defined in the JIS code set table can be used with this encoding method.

**Note:** These coding schemes do not permit simultaneous use of the upper half of the Latin-1 character set.

### A2.3.3 File Naming Rules

The filename is determined by the name of the unit the file contains. The name is formed by taking the full expanded name of the unit and replacing the separating dots with hyphens and using lower case for all letters except that a hyphen in the second character position is replaced by a plus sign.

The extension is `.ads` for a spec and `.adb` for a body as shown in the following table.

<code>`main.ads'</code>	<b>Main (spec)</b>
<code>`main.adb'</code>	<b>Main (body)</b>
<code>`arith_functions.ads'</code>	<b>Arith_Functions (package spec)</b>
<code>`arith_functions.adb'</code>	<b>Arith_Functions (package body)</b>
<code>`func-spec.ads'</code>	<b>Func.Spec (child package spec)</b>
<code>`func-spec.adb'</code>	<b>Func.Spec (child package body)</b>
<code>`main-sub.adb'</code>	<b>Sub (subunit of Main)</b>

Following these rules can result in very long filenames if corresponding unit names are very long (for example, if child units or subunits are heavily nested). An option is available to shorten such long filenames (called filename “krunching”). This may be particularly useful when programs being developed with GNAT are to be used on operating systems such as MS-DOS with limited filename lengths. See section “A2.10.2 Using *gnatk8*.”

Of course, no file shortening algorithm can guarantee uniqueness over all possible unit names; if filename krunching is used it is your responsibility to ensure no name clashes occur.

## A2.3.4 Naming of GNAT Source Files

If you want to examine the workings of the GNAT system, the following brief description of its organization may be helpful:

- Files with prefix 'sc' contain the lexical scanner.
- All files prefixed with 'par' are components of the parser. The numbers correspond to chapters of the Ada standard. For example, parsing of select statements can be found in 'par-ch9.adb'.
- All files prefixed with 'sem' perform semantic analysis. The numbers correspond to chapters of the Ada standard. For example, all issues involving context clauses can be found in 'sem\_ch10.adb'.
- All files prefixed with 'exp' perform AST normalization and expansion, using the same numbering scheme. For example, the construction of record initialization procedures is done in 'exp\_ch3.adb'.
- The files prefixed with 'bind' implement the binder, which verifies the consistency of the compilation, determines an order of elaboration, and generates the bind file.
- The files 'atree.ads' and 'atree.adb' detail the low-level data structures used by the front-end.
- The files 'sinfo.ads' and 'sinfo.adb' detail the structure of the abstract syntax tree as produced by the parser.
- The files 'einfo.ads' and 'einfo.adb' detail the attributes of all entities, computed during semantic analysis.
- Library management issues are dealt with in files with prefix 'lib'.
- Ada files with the prefix 'a-' are children of Ada, as defined in Annex A.
- Files with prefix 'i-' are children of Interfaces, as defined in Annex B.
- Files with prefix 's-' are children of System. This includes both language-defined children and GNAT runtime routines.
- Files with prefix 'g-' are children of GNAT. These are useful general-purpose packages, fully documented in their specifications. All the other '.c' files are modifications of common *gcc* files.

## A2.3.5 Generating Object Files

An Ada program consists of a set of source files, and the first step in compiling the program is to generate the corresponding object files. These are generated by compiling a subset of these source files. The files you need to compile are the following:

- If a package spec has no body, compile the package spec to produce the object file for the package.
- If a package has both a spec and a body, compile the body to produce the object file for the package. The source file for the package spec need not be compiled in this case since there is only one object file, which contains the code for both the spec and body of the package.
- For a subprogram, compile the subprogram body to produce the object file for the subprogram. The spec, if one is present, is as usual in a separate file, and need not be compiled.
- In the case of subunits, only compile the parent unit. A single object file is generated for the entire subunit tree, which includes all the subunits.
- Compile child units completely independently from their parent units (though, of course, the spec of the parent unit must be present).
- Do not compile Generic units (specs and bodies).

The preceding rules describe the set of files that must be compiled to generate the object files for a program. Each object file has the same name as the corresponding source file, except that the extension is `.o` as usual.

You may wish to compile other files for the purpose of checking syntactic and semantic correctness. For example, in the case where a package has a separate spec and body, you would not normally compile the spec. However, it is convenient in practice to compile the spec to make sure it is correct before compiling clients of this spec, since such compilations will fail if there is an error in the spec.

GNAT provides the option for compiling such files purely for the purposes of checking correctness; such compilations are not required as part of the process of building a program.

## A2.3.6 Source Dependencies

A given object file clearly depends on the source file which is compiled to produce it. Here we are using **depends** in the sense of the Unix *make* utility; in other words, an object file depends on a source file if changes to the source file require the object file to be recompiled.

In addition to this basic dependency, a given object may depend on additional source files as follows:

- If a file being compiled *with's* a unit X, the object file depends on the file containing the spec of unit X. This includes files that are *with'ed* implicitly either because they are parents of *with'ed* child units or they are runtime units required by the language constructs used in a particular unit.
- If a file being compiled instantiates a library level generic unit, the object file depends on both the spec and body files for this generic unit.
- If a file being compiled instantiates a generic unit defined within a package, the object file depends on the body file for the package as well as the spec file.
- If a file being compiled contains a call to a subprogram for which `pragma Inline` applies and inlining is activated with the '-gnatn' switch, the object file depends on the file containing the body of this subprogram as well as on the file containing the spec.

The object file for a parent unit depends on the body files for all subunits of the parent unit.

These rules are applied transitively: if unit A *with's* unit B, whose elaboration calls an inlined procedure in package C, the object file for unit A will depend on the body of C, in file 'c.adb'.

The set of dependent files described by these rules includes all the files on which the unit is semantically dependent, as described in the Ada 95 Language Reference Manual. However it is larger because of the inclusion of generic, inline, and subunit dependencies.

An object file must be recreated by recompiling the corresponding source file if any of the source files on which it depends are modified. For example, if the *make* utility is used to control compilation, the rule for an Ada object file must mention all the source files on which the object file

depends.

## A2.3.7 The Ada Library Information Files

Each compilation actually generates two output files. The first of these is the normal object file with a `.o` extension. The second is a text file containing the dependency information file. It has the same name but with an `.ali` extension. This file is known as the Ada Library Information (ALI) file.

You normally need not be concerned with the contents of this file, but this section is included in case you want to understand how these files are being used. Each ALI file consists of a series of lines of the form:

*Key\_Character parameter parameter ...*

The first two lines in the file identify the library output version and Standard version. These are required to be consistent across the entire set of compilation units in your program.

V "xxxxxxxxxxxxxxxxxxx"

This line indicates the library output version, as defined in `gnatvsn.ads`. It ensures that separate object modules of a program are consistent. It must be changed if anything changes that would affect successful binding of modules compiled separately.

Examples of such changes are modifications in the format of the library information described in this package, modifications to calling sequences, or to the way data is represented.

S "xxxxxxxxxxxxxxxxxxx"

This line contains information regarding types declared in packages `Standard` as stored in `Gnatvsn.Standard_Version`.

The purpose (on systems where, for example, the size of `Integer` can be set by command line switches) is to ensure that all units in a program are compiled with a consistent set of options.

The following line is present only for a unit that can be a main program. It has the form:

*M type [priority]*

*type* is either `P` for a parameterless procedure or `F` for a function returning a value of integral type. The latter is for writing a main program that returns an exit status. *priority* is present only if there was a valid `pragma Priority` in the corresponding unit to set the main task priority. It is an unsigned decimal integer.

Following these header lines, a set of information lines appears for each compilation unit that appears in the corresponding object file. In particular, when a package body or subprogram body is compiled there will be two sets of information, one for the spec and one for the body, with the entry for the body appearing first. This is the only case in which a single ALI file contains more than one unit. Note that subunits do not count as compilation units for this purpose, and generate no library information, since they are inlined.

The lines for each compilation unit have the following form:

*U unit-name source-name version [attributes]*

This line identifies the unit to which this section of the library information file applies. *unit-name* is the unit name in internal format, as described in package `Uname`, and *source-file* is the name of the source file containing the unit.

*version* is the version given as eight hexadecimal characters with lower case letters. This value is a hash code that includes contributions from the time stamps of this unit and all its semantically dependent units.

The optional *attributes* are a series of two-letter codes indicating information about the unit:

EB	Unit has pragma Elaborate_Body.
NE	Unit has no elaboration routine. All subprogram specs are in this category, as are subprogram bodies if access before elaboration checks are being generated. Package bodies and specs may or may not have NE set, depending on whether or not elaboration code is required.
PK	Unit is a package, rather than a subprogram.
PU	Unit has pragma Pure.
PR	Unit has pragma Preelaborate.
RC	Unit has pragma Remote_Call_Interface.
RT	Unit has pragma Remote_Types.
SP	Unit has pragma Shared_Passive.
SU	Unit is a subprogram, rather than a package.

The attributes may appear in any order, separated by spaces. Another line in the ALI file has the following form:

```
W unit-name [ source-name lib-name [ E ] [ EA ] ]
```

One of these lines is present for each unit mentioned in an explicit `with` clause by the current unit. *unit-name* is the unit name in internal format. *source-name* is the filename of the file that must be compiled to compile that unit (usually the file for the body, except for packages that have no body). *lib-name* is the filename of the library information file that contains the results of compiling the unit. The E and EA parameters are present if pragma Elaborate or pragma Elaborate\_All, respectively, apply to this unit. In the case of generic units, only *unit-name* is present, since generic units do not need to be compiled, and generate no library information. Note that the elaborate pragmas can be given for generic units, but GNAT ignores them.

Following the unit information is an optional series of lines that indicate the usage of pragma Library\_Unit. For each appearance of pragma Library\_Unit in any of the units for which unit lines are present, a line of the form:

```
L string
```

appears where *string* is the string from the pragma enclosed in quotes. Within the quotes, the following can occur:



- 7-bit graphic characters other than " or {
- " " (indicating a single " character)
- {hh} indicating a character whose code is hex hh

For further details, see `Stringt.Write_String_Table_Entry` in the file `'stringt.ads'`. Note that wide characters in the form `{hhhh}` cannot be produced, since `pragma Linker_Option` accepts only `String`, not `Wide_String`.

Finally, at the end of the ALI file is a series of lines that indicate the source files on which the compiled units depend. This is used by the binder for consistency checking and look like:

```
D source-name time-stamp [ comments]
```

*comments*, if present, must be separated from the time stamp by at least one blank. Currently this field is unused.

Blank lines are ignored when the library information is read, and separate sections of the file are separated by blank lines to ease readability. Extra blanks between fields are also ignored.

## A2.3.8 Representation of Time Stamps

All compiled units are marked with a time stamp, which is derived from The source file. The binder uses these time stamps to ensure consistency of the set of units that constitutes a single program. Time stamps are twelve-character strings of the form `YYMMDDHHMMSS`. Each two-character field has the following meaning:

<code>YY</code>	year	(2 low order digits)
<code>MM</code>	month	(2 digits 01-12)
<code>DD</code>	day	(2 digits 01-31)
<code>HH</code>	hour	(2 digits 00-23)
<code>MM</code>	minutes	(2 digits 00-59)
<code>SS</code>	seconds	(2 digits 00-59)

Time stamps may be compared lexicographically (in other words, the order of Ada comparison operations on strings) to determine which is later or earlier. However, in normal mode, only equality comparisons have any effect on the semantics of the library. Later/earlier comparisons are used only for determining the most informative error messages to be issued by the binder.

The time stamp is the actual stamp stored with the file without any adjustment resulting from time zone comparisons. This avoids problems in using libraries across networks with clients spread across multiple time zones, but may mean the time stamp will differ from that displayed in a directory listing. For example, in UNIX systems, file time stamps are stored in Greenwich Mean Time (GMT), but the *ls* command displays local times.

### A2.3.9 Binding an Ada Program

When using languages such as C and C++, the only remaining step in building an executable program once the source files have been compiled is linking the object modules together. This means it is possible to link an inconsistent version of a program in which two units have included different versions of the same header.

The rules in Ada do not permit such an inconsistent program to be built. For example, if two clients have different versions of the same package, it is not possible to build a program containing these two clients. These rules are enforced by the GNAT binder, which also determines an elaboration order consistent with the Ada rules.

The GNAT binder is run after all the object files for a program have been compiled. It is given the name of the main program unit, and from this it determines the set of units required by the program, reading the corresponding ALI files. It generates error messages if the program is inconsistent or if no valid order of elaboration exists.

If no errors are detected, the binder produces a main program, in C, that contains calls to the required elaboration procedures, followed by a call to the main program. This C program is compiled using the C compiler to generate the object file for the main program. The name of the C file is *b\_XXX.c* where *XXX* is the name of the main program unit.

Finally, the linker is used to build the resulting executable program, using the object from the main program from the bind step as well as the object files for the Ada units of the program.

## A2.3.10 Mixed Language Programming

You build a program that contains some Ada files and some other language files in one of two ways, depending on whether the main program is in Ada or not.

If the main program is in Ada, you proceed as follows:

1. Compile the Ada units to produce a set of object files and ALI files.
2. Compile the other language files to generate object files.
3. Run the Ada binder on the Ada main program.
4. Compile the Ada main program.
5. Link the Ada main program, Ada objects and other language objects

If the main program is in some language other than Ada, you use a special option of the binder to generate callable routines to initialize and finalize the Ada units. You must insert calls to these routines in the main program, or some other appropriate point. The call to initialize the Ada units must occur before the first Ada subprogram is called, and the call to finalize the Ada units must occur after the last Ada subprogram returns. You use the same procedure for building the program as described previously. In this case, however, the binder places the initialization and finalization subprograms into file `'b_XXX.c'` instead of the main program.

## A2.3.11 Comparison of GNAT Model With C/C++ Compilation Model

The GNAT model of compilation is close to the C and C++ models. You can think of Ada specs as corresponding to header files in C. As in C, you don't need to compile specs; they are compiled when they are used. The Ada `with` is similar in effect to the `#include` of a C header.

One notable difference is that, in Ada, you may compile specs separately to check them for semantic and syntactic accuracy. This is not always possible with C headers because they are fragments of programs that have no specific syntactic or semantic rules.

The other major difference is the requirement for running the binder, which performs two important functions. First, it checks for consistency. In C or C++, the only defense against putting together inconsistent programs is outside the compiler, in a make file, for example. The binder satisfies the Ada requirement that it be impossible to construct an inconsistent program when the compiler is used in normal mode.

The other important function of the binder is to deal with elaboration issues. There are also elaboration issues in C++ that are handled automatically. This automatic handling has the advantage of being simpler to use, but the C++ programmer has no control over elaboration. Where gnatbind might complain there was no valid order of elaboration, a C++ compiler would simply construct a program that malfunctioned at runtime.

## A2.3.12 Comparison of GNAT Model With Traditional Ada Library Model

This section is intended to be useful to Ada programmers who have previously used an Ada compiler implementing the traditional Ada library model, as described in the Ada 95 Languages Reference Manual. If you have not used such a system, please go on to the next section.

In GNAT, there is no **library** in the normal sense. Instead, the set of source files themselves acts as the library. Compiling Ada programs does not generate any centralized information, but rather an object file and a ALI file, which are of interest only to the binder and linker.

In a traditional system, the compiler reads information not only from the source file being compiled, but also from the centralized library. This means that the effect of a compilation depends on what has been previously compiled. In particular:

- When a unit is `with`'ed, the unit seen by the compiler corresponds to the version of the unit most recently compiled into the library.
- Inlining is effective only if the necessary body has already been compiled into the library.
- Compiling a unit may obsolete other units in the library.

In GNAT, compiling one unit never affects the compilation of any other units since the compiler reads only source files. Only changes to source files can affect the results of a compilation. In particular:

- When a unit is `with`'ed, the unit seen by the compiler corresponds to the source version of the unit that is currently accessible to the compiler.
- Inlining requires the appropriate source files for the package or subprogram bodies to be available to the compiler. Inlining is always effective, independent of the order in which units are compiled.
- Compiling a unit never affects any other compilations. The editing of sources may cause previous compilations to be out of date if they depended on the source file being modified.

The important result of these differences are that order of compilation is never significant in GNAT. There is no situation in which you are required to do one compilation before another. What shows up as order of compilation requirements in the traditional Ada library becomes, in GNAT, simple source dependencies; in other words, it shows up as a set of rules saying what source files must be present when a file is compiled.

## A2.4.0 Compiling Ada Programs With *gcc*

This chapter discusses how to compile Ada program and the switches passed to the compiler.

### A2.4.1 Compiling Programs

The first step in creating an executable program is to compile the units of the program using the *gcc* command. You must compile the following files:

- the body file (*.adb*) for a library level subprogram
- the spec file (*.ads*) for a library level package that has no body
- the body file (*.adb*) for a library level package that does have a body

You need *not* compile the following files

- the body or spec of a generic library unit
- the spec of a library unit which has a body
- subunits

because they are compiled as part of compiling related units. GNAT compiles generic units when a client instantiates the generic, specs when the corresponding body is compiled, and subunits when the parent is compiled.

If you attempt to compile any of these files, you will get this error message:

```
No code generated for unit xxx in file yyy
```

The basic command for compiling a file containing an Ada unit is:

```
gcc -c [switches] filename
```

where *filename* is the name of the Ada file (having an extension *.ads* for a spec or *.adb* for a body). You specify the *-c* switch to tell *gcc* to compile, but not link, the file. The result of a successful compilation is an object file, which has the same

name as the source file but an extension of `.o` and an Ada Library Information (ALI) file, which also has the same name as the source file, but with `.ali` as the extension. GNAT creates these two output files in the current directory, but you may specify a source file in any directory using an absolute or relative path specification containing the directory information.

*switches* consist of standard *gcc* switches, as documented in the *gcc* manual, as well as GNAT specific switches, which always start with `-gnat`. You may specify these switches in any order, and, in particular, may mix *gcc* and GNAT switches freely.

*gcc* is actually a driver program that looks at the extensions of the file arguments and loads the appropriate compiler. For example, the GNU C compiler is `cc1`, and the Ada compiler is `gnat1`. These programs are in directories known to the driver program (in some configurations via environment variables you set), but need not be in your path. The *gcc* driver also calls the assembler and any other utilities needed to complete the generation of the required object files.

It is possible to supply several filenames on the same *gcc* command. This causes *gcc* to call the appropriate compiler for each file. For example, the command:

```
gcc -c x.adb y.adb z.c
```

calls *gnat1* (the Ada compiler) twice to compile `x.adb` and `y.adb`, and *cc1* (the C compiler) once to compile `z.c`. The compiler generates three object files `x.o`, `y.o` and `z.o` and the two ALI files `x.ali` and `y.ali` from the Ada compilations. Any switches apply to all the files, except for `-gnatx` switches, which apply only to Ada compilations.

## A2.4.2 Switches for *gcc*

The *gcc* command accepts numerous switches. The following are the ones you will most likely need. Note that these switches are case sensitive.

`-b target`

Compile your program to run on target, which is the name of a system configuration. You must have a GNAT cross-compiler built if target is not the same as your host system.

``-Bdir'`

Load compiler executables (for example, *gnat1*, the Ada compiler) from *dir* instead of the default location. Only use this switch when multiple versions of the GNAT compiler are available. See the *gcc* manual page for further details. You would normally use the `'-b'` or `'-v'` switch instead.

``-c'`

Compile. Always use this switch when compiling Ada programs. Note that you may not use *gcc* without a `'-c'` switch to compile and link in one step. This is because the binder must be run, and currently *gcc* cannot be used to run the GNAT binder.

``-g[options]'`

Generate debugging information. This information is stored in the object file and copied from there to the final executable file by the linker, where it can be read by the debugger. You must use the `'-g'` switch if you plan on using the debugger. *options* are used to choose the debugging format if more than one is available; see the *gcc* manual for more details.

``-Idir'`

Direct GNAT to search the *dir* directory for source files needed by the current compilation (see section “A2.4.4 Search Paths and the Run-Time Library (RTL)”).

``-o file'`

This switch is used in *gcc* to redirect the generated object file. Do not use it with GNAT, because it causes the object file and ALI file to have different names and locations. These files should always be kept together.

``-O[n]'`

*n* controls the optimization level.

*n* = 0 No optimization, the default setting if no `'-O'` appears

*n* = 1 Normal optimization, the default if you specify `'-O'` without an operand.

*n* = 2 Extensive optimization

*n* = 3 Extensive optimization with automatic inlining. This applies only to inlining within a unit. See section “A2.4.3.10 Subprogram Inlining Control” for details on control of inter-unit inlining.



``-S'`

Use in place of ``-c'` to cause the assembler source file to be generated, using `'.s'` as the extension, instead of the object file. This may be useful if you need to examine the generated assembly code.

``-v'`

Show commands generated by the `gcc` driver. Normally used only for debugging purposes or if you need to be sure what version of the compiler you are executing.

``-V ver'`

Execute `ver` version of the compiler. This is the `gcc` version, not the GNAT version.

`'-Wuninitialized'`

Generate warnings for uninitialized variables. You must also specify the `'-O'` switch (in other words, `'-Wuninitialized'` works only if optimization is turned on).

Many additional `gcc` switches are relevant. They are fully described in the `gcc` manual.

### A2.4.3 Switches for GNAT

Below is a brief summary of the switches accepted by the `gcc` command when compiling GNAT programs. It is followed by a more complete description of these switches, which has been organized functionally.

<code>-gnata</code>	Assertions enabled. <code>Pragma Assert</code> and <code>pragma Debug</code> to be activated.
<code>-gnatb</code>	Generate brief messages to <code>stderr</code> even if verbose mode set.
<code>-gnatc</code>	Check syntax and semantics only (no code generation attempted).
<code>-gnate</code>	Error messages generated immediately, not saved up till end.
<code>-gnatf</code>	Full errors. Multiple errors per line, all undefined references.
<code>-gnatg</code>	GNAT style checks enabled.
<code>-gnatic</code>	Identifier char set ( <code>c=1/2/3/4/8/p/f/n/w</code> ).
<code>-gnatje</code>	Wide character encoding method ( <code>e=n/h/u/s/e</code> ).
<code>-gnatkn</code>	Limit filenames to <code>n</code> (1-999) characters ( <code>'k' = krunch</code> ).
<code>-gnatl</code>	Output full source listing with embedded error messages.

-gnatmn	Limit number of detected errors to <i>n</i> (1-999).
-gnatn	Activate inlining across unit boundaries.
-gnato	Enable other checks, not normally enabled by default, including numeric overflow checking, and access before elaboration checks.
-gnatp	Suppress all checks.
-gnatq	Don't quit; try semantics, even if parse errors.
-gnatr	Reference manual column layout required.
-gnats	Syntax check only.
-gnatt	Tree output file to be generated.
-gnatu	List units for this compilation.
-gnatv	Verbose mode. Full error output with source lines to stdout.
-gnatwm	Warning mode. ( <i>m=s/e</i> for suppress/treat as error).
-gnatzm	Distribution stub generation ( <i>m=r/s</i> for receiver/sender stubs).
-gnat83	Enforce Ada 83 restrictions.

You may combine a sequence of GNAT switches into a single switch. For example, the specifying the switch

```
-gnatcfi3
```

is equivalent to specifying the following sequence of switches:

```
-gnatc -gnatf -gnati3
```

### A2.4.3.1 Error Message Control

The standard default format for error messages is called "brief format." Brief format messages are written to `stdout` (the standard output file) and have the following form:

```
e.adb:3:04: Incorrect spelling of keyword "function"
e.adb:4:20: ";" should be "is"
```

The first integer after the filename is the line number and the second integer is the column number. *emacs* can parse the error messages and point to the referenced character.

The following '-gnat' switches allow control over the error message format:

'-gnatv'

The **v** stands for verbose. The effect is to write long-format error messages to `stdout`. The same program compiled with the '-gnatv' switch would generate:

```
3. function X (Q : Integer)
   |
>>> Incorrect spelling of keyword "function"
4. return Integer;
   |
>>> ";" should be "is"
```

The vertical bar indicates the location of the error, and the '>>>' prefix can be used to search for error messages. When this switch is used the only source lines output are those with errors.

'-gnatl'

The **l** stands for list. '-gnatl' causes a full listing of the file to be generated. The output is as follows:

```
1. procedure E is
2.   V : Integer;
3.   function X (Q : Integer)
   |
>>> Incorrect spelling of keyword "function"
4.   return Integer;
   |
>>> ";" should be "is"
5.   begin

6.     return Q + Q;
7.   end;
8. begin
9.   V := X + X;
10. end E;
```

When you specify the '-gnatv' or '-gnatl' switches and standard output is redirected, a brief summary is written to `stderr` (standard error) giving the number of error messages and warning messages generated.

`'-gnatb'`

The 'b' stands for brief. This switch causes GNAT to generate the brief format error messages to stdout as well as the verbose format message or full listing.

`'-gnatmn'`

The 'm' stands for maximum. n is a decimal integer in the range of 1 to 999 and limits the number of error messages to be generated. For example, using `'-gnatm2'` might yield

```
e.adb:3:04: Incorrect spelling of keyword "function"  
e.adb:5:35: missing ".."  
fatal error: maximum errors reached  
compilation abandoned
```

`'-gnatf'`

Normally, the compiler suppresses error messages that are likely to be redundant. This switch, where 'f' stands for full, causes all error messages to be generated. One particular effect is for the case of references to undefined variables. If a given variable is referenced several times, the normal format of messages is

```
e.adb:7:07: "V" is undefined (more references follow)
```

where the parenthetical comment warns that there are additional references to the variable V. Compiling the same program with the `'-gnatf'` switch yields:

```
e.adb:7:07: "V" is undefined  
e.adb:8:07: "V" is undefined  
e.adb:8:12: "V" is undefined  
e.adb:8:16: "V" is undefined  
e.adb:9:07: "V" is undefined  
e.adb:9:12: "V" is undefined
```

`'-gnatq'`

In normal operation mode the compiler first parses the program and determines if there are any syntax errors. If there are, appropriate error messages are generated and compilation is immediately terminated. Here 'q' is for quit (really "don't quit") and this switch tells GNAT to continue with semantic analysis even if syntax errors have been found. This may enable the detection of more errors in a single run. On the other hand, the semantic analyzer is more likely to encounter some internal fatal error when given a syntactically invalid tree.

`'-gnate'`

Normally, the compiler saves up error messages and generates them at the end of compilation in proper sequence. This switch (the 'e' stands for error) causes error messages to be generated as soon as they are detected. The use of `'-gnate'` usually causes error messages to be generated out of sequence. Use it when the compiler blows up due to an internal error. In this case, the error messages may be lost. Sometimes blowups are the result of mishandled error messages, so you may want to run with the `'-gnate'` switch to determine whether any error messages were generated (see section "A2.12.2 GNAT Crashes").

In addition to error messages, corresponding to illegalities as defined in the reference manual, the compiler detects two kinds of warning situations.

First, the compiler considers some constructs suspicious and generates a warning message to alert you to a possible error. Second, if the compiler detects a situation that is sure to raise an exception at runtime, it generates a warning message. The following shows an example of warning messages:

```
e.adb:4:24: warning: creation of object of this type may raise
           Storage_Error
e.adb:10:17: warning: static value out of range
e.adb:10:17: warning: "Constraint_Error" will be raised at
runtime
```

Two switches are available to control the handling of warning messages:

`'-gnatwe'`

The 'w' stands for warning and the 'e' stands for error. The `'-gnatwe'` switch causes warning messages to be treated as errors. The warning string still appears, but the warning messages are counted as errors, and prevent the generation of an object file.

`'-gnatws'`

The 's' stands for suppress. This switch completely suppress the output of all warning messages.

### A2.4.3.2 Debugging and Assertion Control

`'-gnata'`

The pragmas `Assert` and `Debug` normally have no effect and are ignored. This switch, where 'a' stands for assert, causes `Assert` and `Debug` pragmas to be activated. The pragmas have the form:

```
pragma Assert (Boolean-expression [, static-string-expression])
pragma Debug (procedure)
```

The `Assert` pragma causes `Boolean-expression` to be tested. If the result is `True`, the pragma has no effect (other than possible side effects from evaluating the expression). If the result is `False`, the exception `System.Assertions.Assert_Error` is raised (passing `static-string-expression`, if present, as the message associated with the exception). The `Debug` pragma causes `procedure` to be called. Note that `pragma Debug` may appear within a declaration sequence, allowing debugging procedures to be called between declarations.

### A2.4.3.3 Runtime Checks

If you compile with the default options, GNAT will insert many runtime checks into the compiled code, including code that performs range checking against constraints, but not arithmetic overflow checking for integer operations (including division by zero) or checks for access before elaboration on subprogram calls. All other runtime checks, as required by the Ada 95 Reference Manual, are generated by default.

The following two `gcc` switches refine this default behavior:

`'-gnatp'`

Suppress all runtime checks as though you have `pragma Suppress (all_checks)` in your source. Use this switch to improve the performance of the code at the expense of safety in the presence of invalid data or program bugs.

`'-gnato'`

Enables overflow checking for integer operations and checks for access before elaboration on subprogram calls ('o' stands for "other checks"). This causes GNAT to generate slower and larger executable programs by adding code to check for both overflow and division by zero (resulting in raising `Constraint_Error` as required by Ada semantics). Similarly, GNAT does not generate elaboration checks by default, and you must specify the `'-gnato'` switch to enable them. Note that the `'-gnato'` switch does not affect the code generated for any floating-point operations; it applies only to integer operations. For floating-point, GNAT has the `Machine_Overflows` attribute set to `False` and the normal mode of operation is to generate IEEE NaN and infinite values on overflow or invalid operations (such as dividing 0.0 by 0.0).

The setting of these switches only controls the default setting of the checks. You may modify them using either `Suppress` (to remove checks) or `Unsuppress` (to add back suppressed checks) pragmas in the program source.

### A2.4.3.4 Using `gcc` for Syntax Checking

`'-gnats'`

Run GNAT in syntax checking only mode ('s' stands for syntax). For example, the command

```
gcc -c -gnats x.adb
```

compiles file `'x.adb'` in syntax-check-only mode. You can check a series of files in a single command, and can use wild cards to specify such a group of files. Note that you must specify the `'-c'` (compile only) flag in addition to the `'-gnats'` flag. You may use other switches in conjunction with `'-gnats'`. In particular, `'-gnatl'` and `'-gnatv'` are useful to control the format of any generated error messages. The output is simply the error messages, if any. No object file or ALI file is generated by a syntax-only compilation. Also, no units other than the one specified are accessed. For example, if a unit `X` with's a unit `Y`, compiling unit `X` in syntax check only mode does not access the source file containing unit `Y`. Normally, GNAT allows only a single unit in a source file. However, this restriction does not apply in syntax-check-only mode, and it is possible to check a file containing multiple compilation units concatenated together. This is primarily used by the `gnatchop` utility (see section "8.0 Handling Files With Multiple Units With `gnatchop`").

### A2.4.3.5 Using `gcc` for Semantic Checking

`'-gnatc'`

Cause the compiler to operate in semantic check mode (`'c'` stands for check), with full checking for all illegalities specified in the reference manual, but without generation of any source code (no object or ALI file generated). Since dependent files must be accessed, you must follow the GNAT semantic restrictions on file structuring to operate in this mode:

- The needed source files must be accessible (see section “A2.4.4 Search Paths and the Run-Time Library (RTL)”).
- Each file must contain only one compilation unit.
- The filename and unit name must match (see section “A2.3.3 File Naming Rules”).

The output consists of error messages as appropriate. No object file or ALI file is generated. The checking corresponds exactly to the notion of legality in the Ada reference manual. Any unit can be compiled in semantics-checking-only mode, including units that would not normally be compiled (generic library units, subunits, and specifications where a separate body is present).

### A2.4.3.6 Compiling Ada 83 Programs

`'-gnat83'`

Although GNAT is primarily an Ada 95 compiler, it accepts this switch to specify that an Ada 83 mode program is being compiled. If you specify this switch, GNAT rejects Ada 95 extensions and applies Ada 83 semantics. It is not possible to guarantee this switch does a perfect job; for example, some subtle tests of pathological cases, such as are found in ACVC tests that have been removed from the ACVC suite for Ada 95, may not compile correctly. However for practical purposes, using this switch should ensure that programs that compile correctly under the `'-gnat83'` switch can be ported reasonably easily to an Ada 83 compiler. This is the main use of the switch. With few exceptions (most notably the need to use `<>` on unconstrained generic formal parameters), it is not necessary to use the `'-gnat83'` switch when compiling Ada 83 programs, because, with rare and obscure exceptions, Ada 95 is upwardly compatible with Ada 83. This means that a correct Ada 83 program is usually also a correct Ada 95 program.



### A2.4.3.7 Style Checking

`'-gnatr'`

Normally, GNAT permits any code layout consistent with the reference manual requirements. This switch (`'r'` is for "reference manual") enforces the layout conventions suggested by the examples and syntax rules of the Ada Language Reference Manual. For example, an `else` must line up with an `if` and code in the `then` and `else` parts must be indented. The compile considers violations of the layout rules a syntax error if you specify this switch.

`'-gnatg'`

Enforces a set of style conventions that correspond to the style used in the GNAT source code. All compiler units are always compiled with the `'-gnatg'` switch specified. You can find the full documentation for the style conventions imposed by `'-gnatg'` in the body of the package `Style` in the compiler sources (in the file `'style.adb'`). You should not normally use the `'-gnatg'` switch. However, you must use `'-gnatg'` for compiling any language-defined unit, or for adding children to any language-defined unit other than `Standard`.

### A2.4.3.8 Character Set Control

`'-gnatic'`

Normally GNAT recognizes the Latin-1 character set in source program identifiers, as described in the reference manual. This switch causes GNAT to recognize alternate character sets in identifiers. `c` is a single character indicating the character set, as follows:

- `'1'` Latin-1 identifiers
- `'2'` Latin-2 letters allowed in identifiers
- `'3'` Latin-3 letters allowed in identifiers
- `'4'` Latin-4 letters allowed in identifiers
- `'p'` IBM PC letters (code page 437) allowed in identifiers
- `'8'` IBM PC letters (code page 850) allowed in identifiers
- `'f'` Full upper-half codes allowed in identifiers
- `'n'` No upper-half codes allowed in identifiers
- `'w'` Wide-character codes allowed in identifiers

See section “A2.3.2 Foreign Language Representation” for full details on the implementation of these character sets.

`'-gnatje'`

Specify the method of encoding for wide characters. *e* is one of the following:

- `'n'` No wide characters allowed (default setting)
- `'h'` Hex encoding
- `'u'` Upper half encoding
- `'s'` Shift/JIS encoding
- `'e'` EUC encoding

### A2.4.3.9 File Naming Control

`'-gnatkn'`

Activates filename “krunching”. *n*, a decimal integer in the range 1-999, indicates the maximum allowable length of a filename (not including the `'ads'` or `'adb'` extension). The default is not to enable filename krunching. For the source file naming rules, see section “A2.3.3 File Naming Rules”.

### A2.4.3.10 Subprogram Inlining Control

`'-gnatn'`

GNAT recognizes and processes `Inline` pragmas. However, for the inlining to actually occur, optimization must be enabled. To enable inlining across unit boundaries, this is, inlining a call in one unit of a subprogram declared in a `with`'ed unit, you must also specify this switch (`'n'` suggests the first syllable of the word “inline”). In the absence of the `'-gnatn'` switch, GNAT does not attempt inlining across units and does not need to access the bodies of subprograms for which `pragma Inline` is specified if they are not in the current unit. If you specify `'-gnatn'`, the compiler will access these bodies, creating an extra source dependency for the resulting object file, and where possible, the call will be inlined. See section “A2.13.3 Inlining of Subprograms” for further details on when inlining is possible.

### A2.4.3.11 Auxiliary Output Control

`'-gnatt'`

Cause GNAT to write the internal tree for a unit to a file (with the extension `' .atb'` for a body or `' .ats'` for a spec). This is not normally required, but is used by separate analysis tools. Typically these tools do the necessary compilations automatically, so you should never have to specify this switch in normal operation.

`'-gnatu'`

Print a list of units required by this compilation on stdout. The listing includes all units on which the unit being compiled depends either directly or indirectly.

### A2.4.3.12 Debugging Control

`'-gnatdx'`

Activate internal debugging switches. `x` is a letter or digit, or string of letters or digits, which specifies the type of debugging outputs desired. Normally these are used only for internal development or system debugging purposes. You can find full documentation for these switches in the body of the `Debug` unit in the compiler source file `' debug.adb'`. One switch you may wish to use is `'-gnatdg,'` which causes a listing of the generated code in Ada source form. For example, all tasking constructs are reduced to appropriate runtime library calls. The syntax of this listing is close to normal Ada with the following additions:

```
new xxx [storage_pool = yyy]
```

Shows the storage pool being used for an allocator.

```
at end procedure-name;
```

Shows the finalization (cleanup) procedure for a scope.

```
(if expr then expr else expr)
```

Conditional expression equivalent to the `x?y:z` construction in C.

```
target^(source)
```

A conversion with floating-point truncation instead of rounding.

*target?(source)*

A conversion that bypasses normal Ada semantic checking. In particular enumeration types and fixed-point types are treated simply as integers.

*target?^(source)*

Combines the above two cases.

*x #/ y*

*x #mod y*

*x #\* y*

*x #rem y*

A division or multiplication of fixed-point values which are treated as integers without any kind of scaling.

*free expr [storage\_pool = xxx]*

Shows the storage pool associated with a free statement.

*freeze typename [actions]*

Shows the point at which *typename* is frozen, with possible associated actions to be performed at the freeze point.

*reference itype*

Reference (and hence definition) to internal type *itype*.

*function-name! (arg, arg, arg)*

Intrinsic function call.

*labelname : label*

Declaration of label *labelname*.

*expr && expr && expr ... && expr*

A multiple concatenation (same effect as *expr & expr & expr*, but handled more efficiently).

*[constraint\_error]*

Raise the `Constraint_Error` exception.

*expression'reference*

A pointer to the result of evaluating *expression*.

*target-type!(source-expression)*

An unchecked conversion of *source-expression* to *target-type*.

*[numerator/denominator]*

Used to represent internal real literals (that) have no exact representation in base 2-16 (for example, the result of compile time evaluation of the expression 1.0/27.0).

## A2.4.4 Search Paths and the Run-Time Library (RTL)

With the GNAT source-based library system, the compiler must be able to find source files for units that are needed by the unit being compiled. Search paths are used to guide this process.

The compiler compiles one source file whose name must be given explicitly on the command line. In other words, no searching is done for this file. To find all other source files that are needed (the most common being the specs of units), the compiler looks in the following directories, in the following order:

1. The directory containing the source file of the main unit being compiled (the filename on the command line).
2. Each directory named by an `'-I'` switch given on the `gcc` command line, in the order given.
3. Each of the directories listed in the value of the `ADA_SOURCE_PATH` environment variable. Construct this value exactly as the `PATH` environment variable: a list of directory names separated by colons.
4. The default location for the GNAT Run Time Library (RTL) source files. This is determined at the time GNAT is built and installed on your system.

The compiler outputs its object files and ALI files in the current working directory.

**Caution:** The object file can be redirected with the `'-o'` switch; however, `gcc` and `gnat1` have not been coordinated on this so the ALI file will not go to the right place. Therefore, you should avoid using the `'-o'` switch.

The packages `Ada`, `System`, and `Interfaces` and their children make up the GNAT RTL, together with the simple `System.IO` package used in the “Hello World” example. The sources for these units are needed by the compiler and are kept together in one directory. Not all of the bodies are needed, but all of the sources are kept together anyway. In a normal installation, you need not specify these directory names when compiling or binding. Either the environment variables or the built-in defaults cause these files to be found.

Besides the assistance in using the RTL, a major use of search paths is in compiling sources from multiple directories. This can make development environments much more flexible.

## A2.4.5 Order of Compilation Issues

If, in our earlier example, there were a spec for the `hello` procedure, it would be contained in the file `'hello.ads'`; yet this file would not need to be explicitly compiled. This is the result of the model we chose to implement library management. Details of the model can be found in file `'gnote1.doc'` in the GNAT sources. Some of the unexpected consequences of the model (unexpected from the point of view of existing Ada compiler systems) are the following:

- There is no point in compiling generics or specs (except for package specs with no bodies) since these are compiled as needed by clients. If you attempt a useless compilation, you will receive an error message. It is also useless to compile subunits since they are compiled as needed by the parent.
- There are no order of compilation requirements and performing a compilation never obsoletes anything. The only way you can obsolete something and require recompilations is to modify one of the dependent source files.
- There is no library as such, apart from the ALI files (see section “A2.3.7 The Ada Library Information Files”, for information on the format of these files). For now we find it convenient to create separate ALI files, but eventually the information therein may be incorporated into the object file directly.
- When you compile a unit, the source files for the specs of all units that it `with's`, all its subunits, and the bodies of any generics it instantiates must be available (findable by the search-paths mechanism described above), or you will receive a fatal error message.

**Note:** The above may seem surprising. However, we are not violating the strict Ada consistency rules; these rules are enforced instead by the binder.

## A2.4.6 Examples

The following are some typical Ada compilation command line examples:

```
gcc -c xyz.adb
```

Compile body in file 'xyz.adb' with all default options.

```
gcc -c -O2 -gnata xyz-def.adb
```

Compile the child unit package in file 'xyz-def.adb' with extensive optimizations, and pragma Assert/Debug statements enabled.

```
gcc -c -gnatc abc-def.adb
```

Compile the subunit in file 'abc-def.adb' in semantic-checking-only mode.

## A2.5.0 Binding Ada Programs With *gnatbind*

This chapter describes the GNAT binder, *gnatbind*, which is used to bind compiled GNAT objects. The *gnatbind* program performs four separate functions:

1. Checks that a program is consistent, in accordance with the rules in chapter 10 of the Ada Language Reference Manual. In particular, error messages are generated if a program uses inconsistent versions of a given unit.
2. Checks that an acceptable order of elaboration exists for the program and issues an error message if it cannot find an order of elaboration satisfying the rules in Chapter 10 of the Ada Language Reference Manual.
3. Generates a main program incorporating the given elaboration order. This program is a small C source file that must be subsequently compiled using the C compiler. The two most important functions of this program are to call the elaboration routines of units in an appropriate order and to call the main program.
4. Determines the set of object files required by the given main program. This information is output as comments in the generated C program, to be read by the *gnatlink* utility used to link the Ada application.

### A2.5.1 Running *gnatbind*

The form of the *gnatbind* command is:

```
gnatbind [switches] mainprog.ali [switches]
```

where *mainprog.adb* is the Ada file containing the main program unit body. If no switches are specified, *gnatbind* constructs a C file whose name is *'b\_mainprog.c'*. For example, if given the parameter *'hello.ali'*, for a main program contained in file *'hello.adb'*, the binder output file would be *'b\_hello.c'*.

When doing consistency checking, the binder takes any source files it can locate into consideration. For example, if the binder determines that the given main



program requires the package `Pack`, whose ALI file is `'pack.ali'` and whose corresponding source spec file is `'pack.ads'`, it attempts to locate the source file `'pack.ads'` (using the same search path conventions as previously described for the `gcc` command). If it can locate this source file, the time stamps must match. In other words, any ALI files mentioning this spec must have resulted from compiling this version of the source file.

The effect of this consistency checking, which includes source files, is that the binder ensures that the program is consistent with the latest version of the source files that can be located at bind time. Editing a source file without compiling files that depend on the source file cause error messages to be generated from the binder.

For example, suppose you have a main program `'hello.adb'` and a package `p`, from file `'p.ads'` and you perform the following steps:

1. Enter `gcc -c hello.adb` to compile the main program.
2. Enter `gcc -c p.ads` to compile package `p`.
3. Edit file `'p.ads'`.
4. Enter `gnatbind hello.ali`.

At this point, the file `'p.ali'` contains an out-of-date time stamp since the file `'p.ads'` has been edited. The attempt at binding fails, and the binder generates the following error messages:

```
error: "hello.adb" must be recompiled ("p.ads" has been
modified)
error: "p.ads" has been modified and must be recompiled
```

Now both files must be recompiled as indicated, and then the bind can succeed, generating a main program. You need not normally be concerned with the contents of this file, but it is similar to the following:

```
int
__main_priority ()
{
    return -1;
}
extern int gnat_argc;
extern char **gnat_argv;
```

```

extern int gnat_exit_status;
void main (argc, argv)
int argc;
char **argv;
{
    gnat_argc = argc;
    gnat_argv = argv;

    __gnat_initialize();
    system__task_specific_data__elabb ();
    p__elabs ();

    _ada_hello ();
    __gnat_finalize();
    exit (gnat_exit_status);
}
unsigned helloB = 0x86c26330;
unsigned system__standard_libraryS = 0x06371136;
unsigned pS = 0x4361339a;
unsigned systemS = 0x430ca9a6;
unsigned system__storage_elementsB = 0xc925fce2;
unsigned system__storage_elementsS = 0x86195344;
unsigned system__task_specific_dataB = 0x924bf9bc;
unsigned system__task_specific_dataS = 0x86195344;
unsigned system__tasking_soft_linksB = 0x0c32a681;
unsigned system__tasking_soft_linksS = 0x86195344;
/* BEGIN Object file/option list
/usr/local/adainclude/system.o
/usr/local/adainclude/s-stoele.o
/usr/local/adainclude/s-taspda.o
/usr/local/adainclude/s-tasoli.o
/usr/local/adainclude/s-stalib.o
p.o
hello.o
END Object file/option list */

```

The `__main_priority` function records the environment task priority. A value of `-1` indicates that the main program has no `pragma Priority`, the normal case.

Next there is code to save the `argc` and `argv` values for later access by the `Ada.Command_Line` package. The variable `gnat_exit_status` saves the exit status set by calls to `Ada.Command_Line.Set_Exit_Status` and is used to return an exit status to the system.

The call to `__gnat_initialize` and the corresponding call at the end of execution to `__gnat_finalize` allow any specialized initialization and finalization code to be hooked in. The default versions of these routines do nothing.

The calls to `system__task_specific_data__elabb` and `p__elabs` perform necessary elaboration of units in the program. In our example, only the library routine `System.Task_Specific_Data` (body) and `P` (spec) required elaboration (the main program `'hello.adb'` did not require any elaboration).

The call to `_ada_hello` is the call to the main program.

The list of unsigned constants gives the version number information. Version numbers are computed by combining time stamps of a unit and all units on which it depends. These values are used for implementation of the `Version` and `Body_Version` attributes.

Finally, a set of comments gives full names of all the object files required to be linked for the Ada component of the program. As seen in the previous example, this list includes the files explicitly supplied and referenced by the user as well as implicitly referenced runtime unit files. The directory names for the runtime units depend on the system configuration.

## A2.5.2 Consistency-Checking Modes

As described in the previous section, by default `gnatbind` checks that object files are consistent with one another and are consistent with any source files it can locate. The following switches can be used to modify this behavior:

`'-s'`

Require source files to be present. In this mode, the binder insists on being able to locate all source files that are referenced and checks their consistency. In normal mode, if a source file cannot be located it is simply ignored. If you specify the `'-s'` switch, a missing source file is an error.

`'-x'`

Exclude source files. In this mode, the binder only checks that ALI files are consistent with one another. Source files are not accessed. The binder runs faster in this mode, and there is still a guarantee that the resulting program is

self-consistent. If a source file has been edited since it was last compiled and you specify the `-x` switch, the binder will not detect that the object file is out of date with the source file.

For most purposes the default mode is appropriate, and this is the mode that is normally used when *gnatmake* or *gnatbind* is used to do the bind operation.

### A2.5.3 Error-Message Control

The following switches provide control over the generation of error messages from the binder:

- `-v` Verbose mode. In the normal mode, brief error messages are generated to `stderr`. If the `-v` switch is present, a header is written to `stdout` and any error messages are directed to `stdout`. All that is written to `stderr` is a brief summary message.
- `-b` Generate brief error messages to `stderr` even if verbose mode is specified. This is relevant only when used together with the `-v` switch.
- `-mn` Limits the number of error messages to `n`, a decimal integer in the range 1-999. The binder terminates immediately if this limit is reached.
- `-ws` Suppress all warning messages
- `-we` Treat any warning messages as fatal errors
- `-t` Ignore time stamp errors. Any time stamp error messages are treated as warning messages. This switch essentially disconnects the normal consistency checking, and the resulting program may have undefined semantics if inconsistent units are present. This means that `-t` should be used only in unusual situations, with extreme care.

## A2.5.4 Output Control

The following switches allow additional control over the output generated by the binder.

- `'-e'` Output complete list of elaboration-order dependencies, showing the reason for each dependency. This output can be rather extensive but may be useful in diagnosing problems with elaboration order. The output is written to `stdout`.
- `'-l'` Output chosen elaboration order. The output is written to `stdout`.
- `'-o file'` Set name of output file to `file` instead of the normal `'b_prog.c'` default. You would normally give `file` an extension of `'c'` since it will be a C source program.
- `'-c'` Check only. Do not generate the binder output file. In this mode the binder performs all error checks but does not generate an output file.

## A2.5.5 Binding for Non-Ada Main Programs

In our description in this chapter so far we have assumed the main program is in Ada and the task of the binder is to generate a corresponding function `main` to pass control to this Ada main program. GNAT also supports the building of executable programs where the main program is not in Ada, but some of the called routines are written in Ada and compiled using GNAT. The following switch is used in this situation:

- `'-n'` No main program. The main program is not in Ada.

In this case, most of the functions of the binder are still required, but instead of generating a main program, the binder generates a file containing the following callable routines:

- `adainit` You must call this routine to initialize the Ada part of the program by calling the necessary elaboration routines. A call to `adainit` is required before the first call to an Ada subprogram.
- `adafinal` You must call this routine to perform any library-level finalization required by the Ada subprograms. A call to `adafinal` is required after the last call to an Ada subprogram, and before the program terminates.

If the `'-n'` switch is given, more than one ALI file may appear on the command line for `gnatbind`. The normal **closure** calculation is performed for each of the specified units. Calculating the closure means finding out the set of units involved by tracing `with` references. The reason it is necessary to be able to specify more than one ALI file is that a given program may invoke two or more quite separate groups of Ada subprograms.

The binder takes the name of its output file from the first specified ALI file, unless overridden by the use of the `'-o'` switch. It will be a C source file, which must be compiled using the C compiler.

## A2.5.6 Summary of Binder Switches

The following are the switches available with `gnatbind`:

<code>'-b'</code>	Generate brief messages to <code>stderr</code> even if verbose mode set.
<code>'-c'</code>	Check only, no generation of binder output file.
<code>'-e'</code>	Output complete list of elaboration-order dependencies.
<code>'-I'</code>	Specify directory to be searched for source and ALI files.
<code>'-l'</code>	Output-chosen elaboration order.
<code>'-mn'</code>	Limit number of detected errors to <i>n</i> (1-999).
<code>'-n'</code>	No main program.
<code>'-o file'</code>	Name the output file (default is <code>'b_XXX.c'</code> ).
<code>'-s'</code>	Require all source files to be present.
<code>'-t'</code>	Ignore time-stamp errors.
<code>'-v'</code>	Verbose mode. Write error messages, header, summary output to <code>stdout</code> .
<code>'-wx'</code>	Warning mode ( <i>x=s/e</i> for suppress/treat as error)
<code>'-x'</code>	Exclude source files (check object consistency only)

You may obtain this listing by running the program `gnatbind` with no arguments.

## A2.5.7 Command-Line Access

The package `Ada.Command_Line` provides access to the command-line arguments and program name. In order for this interface to operate correctly, the two variables

```
int gnat_argc;
char **gnat_argv;
```

are declared in one of the GNAT library routines. These variables must be set from the actual `argc` and `argv` values passed to the main program. With no `'-n'` switch present, `gnatbind` generates the C main program to automatically set these variables. If the `'-n'` switch is used, there is no automatic way to set these variables. If they are not set, the procedures in `Ada.Command_Line` will not be available, and any attempt to use them will raise `Constraint_Error`. If command line access is required, your main program must set `gnat_argc` and `gnat_argv` from the `argc` and `argv` values passed to it.

## A2.5.8 Search Paths for *gnatbind*

The binder must be able to find both the ALI files and the source files. For source files, it follows exactly the same search rules as `gcc` (see section “4.4 Search Paths and the Run-Time Library (RTL)”). Search paths are used for finding the ALI files.

The binder takes the name of an ALI file as its argument and needs to locate other ALI files in its recursive processing. These are found in the following directories in the following order:

1. The current working directory.
2. All directories specified by `'-I'` switches on the `gnatbind` command line, in the order given.
3. Each of the directories listed in the value of the `ADA_OBJECTS_PATH` environment variable. Construct this value the same as the `PATH` environment variable; a list of directory names separated by colons.
4. The default location for the GNAT Run-Time Library (RTL) files, determined when GNAT was built and installed on your system.

For source files accessed by the binder, the search procedure is as described above for object files, except that in step 3 the environment variable `ADA_SOURCE_PATH` is used. The binder generates the bind file (a C language source file) in the current working directory.

The packages `Ada`, `System`, and `Interfaces` and their children make up the GNAT Run-Time Library, together with the package `GNAT` and its children which contain a set of useful additional library functions provided by GNAT. The sources for these units are needed by the compiler and are kept together in one directory. The ALI files and object files generated by compiling the RTL are needed by the binder and the linker and are kept together in one directory, typically different from the directory containing the sources. In a normal installation, you need not specify these directory names when compiling or binding. Either the environment variables or the built-in defaults cause these files to be found.

Besides the assistance in using the RTL, a major use of search paths is in compiling sources from multiple directories. This can make development environments much more flexible.

## A2.5.9 Examples of *gnatbind* Usage

This section contains a number of examples of using the GNAT binding utility *gnatbind*.

```
gnatbind hello.ali
```

The main program *Hello* (source program in *'hello.adb'*) is bound using the standard switch settings. The generated main program is *'b\_hello.c'*. This is the normal, default use of the binder.

```
gnatbind main_program.ali -o mainprog.c -x -e
```

The main program *Main\_Program* (source program in *'main\_program.adb'*) is bound, excluding source files from the consistency checking. A full list of elaboration dependencies is output to `stdout`, and the file *'mainprog.c'* is generated.

```
gnatbind -xe main_program.ali -o mainprog.c
```

This command is exactly the same as the previous example. Switches may appear anywhere in the command line, and single letter switches may be combined into a single switch.



```
gnatbind -n math.ali dbase.ali -o ada-control.c
```

The main program is in a language other than Ada, but calls to subprograms in packages `Math` and `Dbase` appear. This call to *gnatbind* generates the file `'ada-control.c'` containing the `adainit` and `adafinal` routines to be called before and after accessing the Ada subprograms.

## A2.6.0 Linking Ada Programs Using *gnatlink*

This chapter discusses *gnatlink*, a utility program used to link Ada programs and build an executable file. This program is basically a simple process which invokes the Unix linker (via the *gcc* command) with a correct list of object files and library references. *gnatlink* automatically determines the list of files and references for the Ada part of a program. It uses the binder file generated by the binder to determine this list.

### A2.6.1 Running *gnatlink*

The form of the *gnatlink* command is

```
gnatlink [switches] mainprog[.ali] [non-Ada objects]
[linker options]
```

'*mainprog.ali*' references the ALI file of the main program. The '.ali' extension of this file can be omitted. From this reference, *gnatlink* locates the corresponding binder file '*b\_mainprog.c*' and, using the information in this file along with the list of non-Ada objects and linker options, constructs a Unix linker command file to create the executable.

The arguments following '*mainprog.ali*' are passed to the linker uninterpreted. They typically include the names of object files for units written in other languages than Ada and any library references required to resolve references in any of these foreign language units, or in `pragma Import` statements in any Ada units. This list may also include linker switches.

*gnatlink* determines the list of objects required by the Ada program and prepends them to the list of objects passed to the linker. *gnatlink* also gathers any arguments set by the use of `pragma Linker_Options` and adds them to the list of arguments presented to the linker.

## A2.6.2 Switches for *gnatlink*

The following switches are available with the *gnatlink* utility:

- '-o *exec-name*'     *exec-name* specifies an alternative name for the generated executable program. If the '-o' switch is omitted, the executable is called the name of the main unit. So *gnatlink try.ali* creates an executable called 'try'.
- '-v'                 Causes additional information to be output, including a full list of the included object files. This switch option is most useful when you want to see what set of object files are being used in the link step.
- '-g'                 The option to include debugging information causes the C bind file (in other words, '*b\_mainprog.c*') to be compiled with '-g'. In addition, the binder does not delete the '*b\_mainprog.c*' and '*b\_mainprog.o*' files (without '-g', the binder removes these files by default).
- '-gnatlink *name*'   *name* is the name of the linker to be invoked. You normally omit this switch, in which case the default name for the linker is ('*gcc*').

## A2.7.0 The GNAT Make Program *gnatmake*

A typical development cycle when working on an Ada program consists of the following steps:

1. Edit some sources to fix bugs.
2. Add enhancements.
3. Compile all sources affected.
4. Rebind and relink.
5. Test.

The third step can be tricky, because not only do the modified files have to be compiled, but any files depending on these files must also be recompiled. The dependency rules in Ada can be quite complex, especially in the presence of overloading, `use` clauses, generics and inlined subprograms.

*gnatmake* automatically takes care of the third and fourth steps of this process. It determines which sources need to be compiled, compiles them, and binds and links the resulting object files.

Unlike some other Ada make programs, the dependencies are always accurately recomputed from the new sources. The source based approach of the GNAT compilation model makes this possible. This means that if changes to the source program cause corresponding changes in dependencies, they will always be tracked exactly correctly by *gnatmake*.

## A2.7.1 Running *gnatmake*

The *gnatmake* command has the form:

```
gnatmake [-a] [-c] [-f] [-g] [-jn] [-k] [-M] [-o exec-name]
[-n] [-q] [-v]
      [compiler_switch]
      {-Adir} {-aOdir} {-aIdir} {-Idir} {-I-} {-Ldir}
      unit_or_file_name
      {-cargs options} {-bargs options} {-largs options}
```

Here square brackets indicate optional components in the command, and curly brackets indicate a construction that can occur zero or more times.

The only required argument is *unit\_or\_file\_name*, which specifies the compilation unit that is the main program. There are two ways to specify this:

- By giving the lowercase name of the compilation unit (`gnatmake unit`).
- By giving the name of the source containing it (`gnatmake file.adb`).

All *gnatmake* output (except when you specify '-M') is to `stderr`. The output produced by the '-M' switch is send to `stdout`.

## A2.7.2 Switches for *gnatmake*

You may specify any of the following switches to *gnatmake*:

- '-a' Consider all files in the make process, even the GNAT internal system files (for example, the predefined Ada library files). By default, *gnatmake* does not check these files (however, if there is an installation problem, it will be caught when *gnatmake* binds your program). You may have to specify this switch if you are working on GNAT itself. The vast majority of *gnatmake* users never need to specify this switch. By default *gnatmake -a* compiles all GNAT internal files with `gcc -c -gnatg` rather than `gcc -c`.
- '-c' Compile only. Do not perform binding and linking. If the root unit specified by *unit\_or\_file\_name* is not a main unit, this is the default. Otherwise *gnatmake* will attempt binding and linking unless all objects are up to date and the executable is more recent than the objects.
- '-f' Force recompilations. Recompile all sources, even though some object files may be up to date, but don't recompile predefined or GNAT internal files unless the '-a' switch is also specified.
- '-g' Compile with debugging information. Same effect as `-cargs -g`.
- '-jn' Use *n* processes to carry out the (re)compilations. If you have a multiprocessor machine, compilations will occur in parallel. In the event of compilation errors, messages from various compilations might get interspersed (but *gnatmake* will give you the full ordered list of failing compiles at the end). This can at times be annoying. To get a clean list of error messages don't use '-j'.
- '-k' Keep going. Continue as much as possible after a compilation error. To ease the programmers's take in case of compilation errors, the list of sources for which the compile fails is given when *gnatmake* terminates.
- '-M' Check if all objects are up to date. If they are output the object dependences to `stdout` in a form that can be directly exploited in a 'Makefile'. By default, each source file is prefixed with its (relative or absolute) directory name. This name is whatever you specified in the various '-aI' and '-I' switches. If you use `gnatmake -M -q` (see '-q' below), only the source file names, without relative paths, are output. If

you just specify the '-M' switch, dependencies of the GNAT internal system files are omitted. This is typically what you want. If you also specify the '-a' switch, dependencies of the GNAT internal files are also listed. Note that dependencies of the objects in external Ada libraries (see switch '-aLdir' in the following list) are never reported.

- '-n' Don't compile, bind, or link. Output a single command that will recompile an out of date unit, if any. Repeated use of this option, followed by carrying out the indicated compilation, will eventually result in recompiling all required units. If any ALI is missing during the process, *gnatmake* halts and displays an error message.
- '-o exec\_name' Output executable name. The name of the final executable program will be *exec\_name*. If the '-o' switch is omitted the default name for the executable will be the name of the input file without the suffix (Unix systems) or the name of the input file with an '.exe' extension (DOS and OS/2). You may prefix *exec\_name* with a relative or absolute directory path.
- '-q' Quiet. When this flag is not set, the commands carried out by *gnatmake* are displayed.
- '-v' Verbose. Displays the reason for all recompilations *gnatmake* decides are necessary.
- 'gcc switches' The switch '-g' or any upper case switch (other than '-A', or '-L') or switch that is more than one character is passed to gcc (e.g. '-O', '-gnato,' etc.)

#### Source & Library search path switches:

- '-Adir' Equivalent to '-aLdir -aIdir'.
- '-aOdir' When looking for library and object files look also in directory *dir*. The order in which library files search is undertaken is described in section "4.4 Search Paths and the Run-Time Library (RTL)".
- '-aIdir' When looking for source files also look in directory *dir*.
- '-Idir' Equivalent to '-aOdir -aIdir'.
- '-I-' Do not look for source, library or object files in the default directory.

'-Lvar' Add directory *dir* to the list of directories in which the linker will search for libraries. This is equivalent to '-largs -Ldir'.

General compiler, binder or linker switches:

'-cargs options'

Compiler arguments. Without '-cargs', *gnatmake* uses `gcc -c` to perform compilations. Otherwise, *gnatmake* uses `gcc -c c_opts`, where *c\_opts* is a list of parameters including all the options encountered in the set of '-cargs' switches present on the *gnatmake* command line. A given sublist of '-cargs' options is terminated upon encountering another '-cargs', '-bargs' or '-largs'. By default, *gnatmake -a* compiles all GNAT internal files with `gcc -c -gnatg` rather than `gcc -c`.

'-bargs options'

Binder arguments. Without '-bargs', *gnatmake* uses `gnatbind unit.ali` to bind. Otherwise, *gnatmake* uses `gnatbind b_opts unit.ali`. *b\_opts* is akin to *c\_opts*, but is obtained from '-bargs' switches. '-largs options'. Similar to '-bargs', but specifies the options for *gnatlink*. Note that you are not allowed to use the '-o' switch within a '-largs'. Use the '-o' switch directly to *gnatmake* to give a specific name to your executable.

### A2.7.3 Notes on the Command Line

Please note the following with regard to *gnatmake*:

- If you enter `gnatmake file.adb`, where '*file.adb*' is a subunit or body of a generic unit, *gnatmake* recompiles '*file.adb*' (because it finds no ALI) and stops, issuing a warning.
- *gnatmake* does not examine '.o' files, only ALI files. Deleting '.o' files will not force a recompile. You can force everything to be recompiled either by deleting the ALI files or by using the '-f' switch.
- If *gnatmake* finds no ALI files, it recompiles the main program and all other units required by the main program. Thus *gnatmake* can be used for the initial compile, as well as during the re-edit development cycle.



## A2.7.4 How *gnatmake* Works

Generally *gnatmake* automatically performs all necessary recompilations and you don't need to worry about how it works. However, it may be useful to have some basic understanding of the *gnatmake* approach and in particular to understand how it uses the results of previous compilations without incorrectly depending on them.

First a definition: an object file is considered **up to date** if the corresponding ALI file exists and if all the source files listed in the dependency section of this ALI file have time stamps matching those in the ALI file. This means that neither the source file itself nor any files that it depends on have been modified, and hence there is no need to recompile this file.

*gnatmake* works by first checking if the specified main program is up to date. If so, it is done, and no compilations are required. If not, it compiles the main program to build a new ALI file that reflects the latest sources. It examines this ALI file to find all the source files on which the main program depends, and recursively applies the up to date test on all these files.

This process ensures that *gnatmake* only trusts the dependencies in an existing ALI file if they are known to be correct. Otherwise it always recompiles to determine a new, guaranteed accurate set of dependencies. Thus the program is compiled “upside down” from what may be more familiar as the required order of compilation in some other Ada systems. In particular, clients are compiled before the units on which they depend. The ability of GNAT to compile in any order is critical in allowing an order of compilation to be chosen that guarantees that *gnatmake* will recompute a correct set of new dependencies if necessary.

## A2.7.5 Examples of *gnatmake* Usage

```
gnatmake hello.adb
```

Compile all files necessary to bind and link the main program 'hello.adb' (containing unit `Hello`) and bind and link the resulting object files to generate an executable file 'hello'.

```
gnatmake -q Main_Unit -cargs -O2 -bargs -l
```

Compile all files necessary to bind and link the main program unit `Main_Unit` (from file 'main\_unit.adb'). All compilations will be done with optimization level 2 and the order of elaboration will be listed by the binder. *gnatmake* will operate in quiet mode, not displaying commands it is executing.

## A2.8.0 Handling Files With Multiple Units With *gnatchop*

This chapter discusses how to handle files with multiple units by using the *gnatchop* utility.

### A2.8.1 Handling Files With Multiple Units

The basic compilation model of GNAT requires a file submitted to the compiler have only one unit and there must be a strict correspondence between the file name and the unit name.

The *gnatchop* utility allows both of these rules to be relaxed, allowing GNAT to process files which contain multiple compilation units and files with arbitrary filenames. The approach used by *gnatchop* is to read the specified file and generate one or more output files, containing one unit per file and with proper filenames as required by GNAT.

If you want to permanently restructure a set of “foreign” files so that they match the GNAT rules and do the remaining development using the GNAT structure, you can simply use *gnatchop* once, generate the new set of files and work with them from that point on.

Alternatively, if you want to keep your files in the “foreign” format, perhaps to maintain compatibility with some other Ada compilation system, you can set up a procedure where you use *gnatchop* each time you compile, regarding the source files that it writes as temporary files that you throw away.

## A2.8.2 Command Line for *gnatchop*

The *gnatchop* command has the form:

```
gnatchop [-k] [-r] [-s] [-w] filename [directory]
```

The only required argument is the filename of the file to be chopped. There are no restrictions on the form of this filename. The file itself contains one or more Ada files, in normal GNAT format, concatenated together.

When run in default mode, *gnatchop* generates one output file in the current directory for each unit in the file. For example, given a file called 'hellofiles' containing:

```
procedure hello;  
with Text_IO; use Text_IO;  
procedure hello is  
begin  
    Put_Line ("Hello");  
end hello;
```

the command:

```
gnatchop hellofiles
```

generates two files in the current directory, one called 'hello.ads' containing the single line that is the procedure spec, and the other called 'hello.adb' containing the remaining text. The original file is not affected. The generated files can be compiled in the normal manner.

## A2.8.3 Switches for *gnatchop*

*gnatchop* recognizes the following switches:

- '-k' Limit generated filenames to eight characters. This is useful if the resulting set of files is required to be interoperable with systems like MS-DOS which limit the length of filenames.

- '-r' Generate `Source_Reference` pragmas. Use this switch if the output files are regarded as temporary and development is to be done in terms of the original unchopped file. The '-r' switch causes `Source_Reference` pragmas to be inserted into each of the generated files to refer back to the original filename and line number. The result is that all error messages refer back to the original unchopped file. In addition, the debugging information placed into the object file (when the '-g' switch of `gcc` or `gnatmake` is specified) also refers back to this original file so that tools like profilers and debuggers will give information in terms of the original unchopped file.
- '-s' Write a compilation script to standard output containing `gcc` commands to compile the generated files.
- '-w' Overwrite existing filenames. Normally `gnatchop` regards it as a fatal error situation if there is already a file with the same name as a file it would otherwise output. The '-w' switch bypasses this check, and any such existing files will be silently overwritten.

*directory*, if specified, gives the name of the directory to which the output files will be written. If it is not specified, all files are written to the current directory.

## A2.8.4 Examples of *gnatchop* Usage

```
gnatchop -w hello_s.ada ~gnat/ichibiah/files
```

Chops the source file 'hello\_s.ada'. The output files will be placed in the directory '~gnat/ichibiah/files', overwriting any files with matching names in that directory (no files in the current directory are modified).

```
gnatchop -s -r collect
```

Chops the source file 'collect' into the current directory. A compilation script is also generated, and all output files have `Source_Reference` pragmas, so error messages will refer back to the file 'collect' with proper line numbers.

```
gnatchop archive
```

Chops the source file 'archive' into the current directory. One useful application of *gnatchop* is in sending sets of sources around, for example in email messages. The required sources are simply concatenated (for example, using a Unix `cat` command), and then *gnatchop* is used at the

other end to reconstitute the original file names.

## A2.9.0 The Front-End/Cross-Reference Utility *gnatf*

This chapter discusses *gnatf*, a stripped-down version of the GNAT compiler containing only the front end. *gnatf* can preform full syntax and semantic checking and also has a cross-reference analyzer built in that can perform a variety of functions.

### A2.9.1 Overview of *gnatf*

The GNAT system provides a stand-alone tool, *gnatf*, which allows for syntax and semantics checking without any code generation. This is somewhat faster than using `gcc -gnatc`.

The standard GNAT switches that do not concern code generation are still available in *gnatf*. However, they should not be preceded by '-gnat', so to do syntax only checking with *gnatf*, use `gnatf -s file.adb`, not `gnatf -gnats file.adb`.

The real point of *gnatf* is that it contains a cross reference tool, whose goals are:

- Giving precise information about all declared entities (where they are defined and where they are used). This is particularly useful in the Ada emacs mode contained in the GNAT distribution.
- Emitting warnings if an entity is defined but never used or a `with` clause is unnecessary, misplaced or redundant.

## A2.9.2 Command Line of *gnatf*

The *gnatf* command line is of the following form:

```
gnatf [switches] files
```

The effect is similar to a *gcc* command specifying the '-gnatc' (no code generation) switch, although it is somewhat faster, especially if several files are processed at the same type.

## A2.9.3 Compilation Switches

The following compilation switches are similar to the corresponding switches in *gcc*, except that the names are not preceded by '-gnat'. For example, the *gnatf* switch for syntax-only checking is '-s' instead of -gnats. For full details on these switches, see section "A2.4.2 Switches for *gcc*."

- '-b' Generate brief messages to stderr even if verbose mode set
- '-e' Error messages generated immediately, not saved up till end
- '-f' Full errors. Multiple errors/line, all undefined references
- '-g' GNAT style checks enabled
- '-ic' Identifier char set (c=1/2/3/4/8/p/f/n/w)
- '-je' Wide character encoding method (e=n/h/u/s/e)
- '-kn' Limit file names to *n* (1-999) character.
- '-l' Output full source listing with embedded error messages
- '-mn' Limit number of detected errors to *n* (1-999)
- '-q' Don't quit, try semantics, even if parse errors
- '-r' Reference manual column layout required
- '-s' Syntax check only
- '-t' Tree output file to be generated
- '-u' List units for this compilation
- '-v' Verbose mode. Full error output with source lines to stdout
- '-wm' Warning mode. (*m=s/e* for suppress/treat as error)
- '-83' Enforce Ada 83 restrictions



## A2.9.4 Cross-Referencing Switches

The following list contains the descriptions of the cross-referencing flags available with *gnatf*:

- '-x1' Issues warnings for unnecessary, misplaced or redundant *with* clauses. Specifically, a warning message is generated in the following cases:
  - A compilation unit which is *with*'ed but never used (this works with child library units as well).
  - A compilation unit that is *with*'ed in a body (responsible subunit) if the same *with* clause already appears in the specification (responsible specification or body for subunits).
  - A compilation unit that is *with*'ed within a specification but is used only by the body or a subunit.
- '-x2' Issues warnings on unused entities, that is entities that are declared but never used. Note that no warnings are issued for unreferenced entities such as the following:
  - Record fields, since they could be referenced indirectly by an aggregate.
  - Enumeration entities, since they could be referenced indirectly by enumeration ranges:
 

```
for i in Color'First .. Color'Last
```
  - Loop parameters:
 

```
for I in 1 .. 80 loop
  Put ('x');
end loop;
```
- '-x[345]' Generate cross-reference information. The '-x3' switch gives the most succinct cross-referencing information, '-x5' the most comprehensive. The '-x4' switch gives more information than '-x3' but not as much as '-x5'. The information given by switches '-x3' and '-x4' is used in the smart recompilation system currently under development. The '-x5' switch lists all entities defined or used in the analyzed compilation units. It gives the source location of their definition and all their uses in the analyzed units.

- '-x6' The cross-reference output is the same as with '-x5', except that with '-x6', all cross-reference information is stored in the single file 'X.ref', and the entity kind of each cross-referenced entity is also given.

## A2.9.5 Cross Reference Information and Smart Recompilation

The cross reference information gathered by the '-x3' and '-x4' switches is a subset of the information specified by the '-x5' switch. The former information is specifically tailored to the smart recompilation system currently under development. When '-x3' or '-x4' are specified, the cross-referencing tool *gnatf* produces the following information for each compilation unit analyzed. We refer to that unit as *unit* in the following list.

- The full graph of the source files, directly or indirectly loaded as a result of compiling *Unit* along with their time stamp. This graph includes the *with*'ed unit graph rooted at *unit* and also other units automatically loaded by GNAT during code generation (generic bodies, subunits, and bodies of inlined subprograms).
- The list of entities that can be exported from *unit* to other Ada sources along with their line and column of definition and use in *unit*. If *unit* is a subprogram or package specification, the notion of exported entity matches the set of entities listed therein. If *unit* is a package body with no generics or inlined subprograms, then no entities are exported. In general, however, the set of entities exported from *unit* is the set of entities that are needed across compilation units by *gnat* when generating code. Specifically inlined subprogram bodies or generic bodies are always exported, since these are inlined at the point of use or instantiation. The same happens for subunits, which are inlined in the parent unit. The difference between the '-x3' and '-x4' switches is that '-x3' omits all generic bodies or inlined subprograms from the exported entities, while '-x4' includes them. Both '-x3' and '-x4' consider subunits as exported entities. *gnatf* considers only outermost visible entities to be exported. That is, a record or enumeration type may be exported, but its inner fields or enumeration literals are never considered exported entities. It is the same for subprogram parameters and discriminants.

- The list of entities directly imported by *unit* from other Ada sources, along with their lines and columns where they are used in *unit*. The idea of imported entities is derived from the notion of exported entities (what is exported by one unit may be imported by another).

## A2.9.6 File Structure

The cross-referencing file is divided into various sections. There is one section for each compilation unit explicitly requested. We call these units, RUs, for “requested units.” There is also one section for each AU, (auxiliary unit); that is, those compilation units that are implicitly loaded by the compiler, but whose compilation has not been explicitly requested by the user. Specs of withed packages are typical AUs.

All entities exported by RUs (the ‘-x3’ and ‘-x4’ switches) or all entities belonging to RUs (the ‘-x5’ and ‘-x6’ switches) appear in the cross-referencing file(s).

However, only the entities defined in AUs that are imported in RUs appear in the cross-referencing file. Their order is the order of declaration in the source files.

The sections in the cross reference referring to RUs and AUs are respectively denoted:

```
%% unit.ad[sb]
   for an RU
-- unit.ad[sb]
   for an AU
```

**Note:** An entity defined inside a generic and used through a generic instantiation is listed under the cross-referencing section of the generic unit.

## A2.9.7 Example of *gnatf* Usage

```
`test.adb`
```

```
01 with Part1; -- unused
02 with Part2; use Part2;
03 procedure Test is
04
05   Thing : Number;
06   type Client is record
07     Number : Integer;
08     State : Boolean;
09   end record;
10   type Color is (Red, Green); -- unused
11   My_Client : Client;
12
13 begin
14   My_Client.Number := 1;
15   My_Client.State := True;
16   Thing := 20;
17   Thing := Thing + Thing;
18 end;
```

```
`part1.ads`
```

```
01 package Part1 is
02   type Useless is new Integer;
03 end;
```

```
`part2.ads`
```

```
01 package Part2 is
02   type Number is new Integer range 1 .. 1000;
03   The_Number : constant := 42;
04 end;
```

The result of invoking `gnatf -x5 test.adb` is the following (just skim the file `test.xrb`, explanations follow:

**Warnings on stderr (the screen):**

```
test.adb:1:06: warning: "Part1" withed but unused.
test.adb:3:11:
warning: "Test" unused test.adb:10:09: warning: "Color"
unused
`test.xrb'

01 V "SGNAT v1.0 "
02 test.adb 941012154746 2 3
03 part1.ads 941012154531
04 part2.ads 941012154620
05
06 %% test.adb
07 test 3:11
08 thing 5:4
09 {16:4 17:4 17:13 17:21}
10 client 6:9
11 {11:16}
12 client.number 7:7
13 {14:14}
14 client.state 8:7
15 {15:14}
16 color 10:9
17 red 10:19
18 green 10:24
19 my_client 11:4
20 {14:4 15:4}
21
22 -- part1.ads
23 part1 1:9
24 {1:6}
25
26 -- part2.ads
27 part2 1:9
28 {2:6 2:17}
29 number 2:9
30 {5:14}
```

The unit `Test` is the only RU (requested unit). AUs (auxiliary units) are packages `Part1` and `Part2`. First, the graph of the loaded units with their time stamps is given:

```
02 test.adb 941012154746 2 3
03 part1.ads 941012154531
04 part2.ads 941012154620
```

Unit `Test` requires the loading of units `Part1` and `Part2` (the second and third units listed in the inclusion graph).

The entry

```
06 %% test.adb
07 [...]
08 thing 5:4
09 {16:4 17:4 17:13 17:21}
```

means `Thing` is an entity (a variable) defined in line 5 column 4; used in line 16 column 4; and in line 17 columns 4, 13, and 21; in file `'test.adb'`.

The entity `Useless` may be used in units other than `Test`, but that information is not contained in the `'test.xrb'` file because `Test` does not use `Useless`.

## A2.10.0 Filename Krunching With *gnatk8*

This chapter discusses the *gnatk8* filename krunching utility.

### A2.10.1 About *gnatk8*

The normal rule in using GNAT is that the filename must be derived from the unit name. The exact default rule is: Take the unit name and replace all dots by hyphens, except that if such a replacement occurs in the second character position of a name, replace the dot by a plus instead of a hyphen.

The '-gnatk $nn$ ' switch of the compiler activates a "krunching" circuit that limits filenames to  $nn$  characters (where  $nn$  is a decimal integer). This is primarily intended for use on MS-DOS and similar systems where  $nn=8$ , to fit in the 8+3 limitation on filenames found in these systems.

The *gnatk8* utility can be used to determine the krunched name for a given file, when krunched to a specified maximum length.

### A2.10.2 Using *gnatk8*

The *gnatk8* command has the form:

```
gnatk8 name [length]
```

*name* can be an Ada name with dots or the GNAT name of the unit where the dots representing child units or subunit are replaced by hyphens. The only confusion arises if a name ends in '.ads' or '.adb'. *gnatk8* takes this to be an extension if there are no other dots in the name and the whole name is in lower case.

*length* represents the length of the krunched name. The default without any argument given is 8 characters. A length of zero stands for unlimited, in other words no chop except for system files which are always 8.

The output is the krunched name. The output has an extension only if the original argument was a filename with an extension.

### A2.10.3 Krunching Method

The initial filename is determined by the name of the unit that the file contains. The name is formed by taking the full expanded name of the unit and replacing the separating dots with hyphens and using lower case for all letter, except that a hyphen in the second character position is replaced by a plus sign. The extension is *.ads* for a specification and *.adb* for a body.

Krunching does not affect the extension, but the filename is shortened to the specified length by following these rules:

- The name is divided into segments separated by minus or plus signs and underscores and all hyphens, plus signs, and underscores are eliminated. If this leaves the name short enough, we are done.
- If not the longest segment is located (left-most if there are two of equal length), and shortened by dropping its last character. This is repeated until the name is short enough. As an example, consider the krunch of *'our-strings-wide\_fixed.adb'* to fit the name into 8 characters as required by MS-DOS:

```
our-strings-wide_fixed 22
our strings wide fixed 19
our string wide fixed 18
our stri wide fixed 17
our stri wide fixe 16
our stri wide fixe 14
our str wide fixe 14
our str wid fixe 13
our str wid fix 12
ou str wid fix 11
ou st wid fix 10
ou st wi fix 9
ou st wi fi 8
Final filename: oustwifi.adb
```



- The filenames for all predefined units are always krunched to eight characters. The krunching of these predefined units uses the following special prefix replacements:

```

`ada-`      replaced by `a-`
`gnat-`    replaced by `g-`
`interfaces-` replaced by `i-`
`system-`  replaced by `s-`

```

These system files have a hyphen in the second character position. That is why normal user files replace such a character with a plus sign, to avoid confusion with system filenames. As an example of this special rule, consider 'ada-strings-wide\_fixed.adb', which gets krunched as follows:

```

ada-strings-wide_fixed 22
a- strings wide fixed 18
a- string  wide fixed 17
a- strin   wide fixed 16
a- stri    wide fixed 15
a- stri    wide fixe  14
a- str     wide fixe  13
a- str     wid  fixe  12
a- str     wid  fix   11
a- st      wid  fix   10
a- st      wi   fix   9
a- st      wi   fi    8
Final filename: a-stwifx.adb

```

Of course no file shortening algorithm can guarantee uniqueness over all possible unit names, and if filename krunching is used then it is your responsibility to ensure that no name clashes occur. The utility program *gnatk8* is supplied for conveniently determining the krunched name of a file.

#### A2.10.4 Examples of *gnatk8* Usage

```

gnatk8 very_long_unit_name.ads      ----> velounna.ads
gnatk8 very_long_unit_name.ads 6    ----> vlunna.ads
gnatk8 very_long_unit_name.ads 0    ----> long_unit_name.ads
gnatk8 grandparent-parent-child.ads ----> grparchi.ads
gnatk8 grandparent.parent.child     ----> grparchi

```

## A2.11.0 Other Utility Programs

This chapter discusses some other utility programs available in the Ada environment.

### A2.11.1 Using Other Utility Programs With GNAT

The object files generated by GNAT are in standard system format and in particular the debugging information uses this format. This means programs generated by GNAT can be used with existing utilities that depend on these formats.

In general, any utility program that works with C will also work with Ada programs generated by GNAT. This includes software utilities such as *gprof* (a profiling program), *gdb* (the FSF debugger), and utilities such as *Purify*.

### A2.11.2 The Naming Scheme of GNAT

In order to interpret the output from GNAT, it is necessary to understand the conventions used to generate link names from the Ada entity names.

All names are in all lower-case letters. With the exception of library procedure names, the mechanism used is simply to use the full expanded Ada name with dots replaced by double underscores. For example, suppose we have the following package spec:

```
package QRS is MN : Integer; end QRS;
```

The variable `MN` has a full expanded Ada name of `QRS.MN`, so the corresponding link name is `qrs__mn`.

Of course if a `pragma Export` is used this may be overridden:

```
package Exports is
  Var1 : Integer;
  pragma Export (Var1, C, External_Name =>
    "var1_name");
  Var2 : Integer;
  pragma Export (Var2, C, Link_Name =>
    "var2_link_name");
end Exports;
```

In this case, the link name for `Var1` is `var1_name`, and the link name for `Var2` is `var2_link_name`.

One exception occurs for library level procedures. A potential ambiguity arises between the required name `_main` for the C main program, and the name we would otherwise assign to an Ada library level procedure called `Main` (which might well not be the main program).

To avoid this ambiguity, we attach the prefix `_ada_` to such names. So if we have a library level procedure such as

```
procedure Hello (S : String);
```

the external name of this procedure will be `_ada_hello`.

### A2.11.3 Ada Mode for Emacs

In the subdirectory `'emacs-ada-mode'` you will find a set of files implementing an Ada mode under GNU emacs. The mode is still under development, but a number of features are complete.

For instance, the Ada mode has the same indenting friendliness that C programmers get with the c-mode, you can toggle between specification and body with a few keystrokes, etc. This mode also uses `'gnatf'` to be able to point to an entity with the mouse, click it and open a window with its definition. This mode is copyrighted by Markus Heritsch and Rolf Ebert.

## A2.12.0 Running and Debugging Ada Programs

This chapter discusses how to run and debug Ada programs.

### A2.12.1 Getting Internal Debugging Information

Most compilers have secret internal debugging switches and modes. GNAT does also, except GNAT internal debugging switches and modes are not secret. A summary and full description of all the compiler and binder debug flags are in the file *'debug.adb'*. You must obtain the sources of the compiler to see the full detailed effects of these flags.

The switches that print the source of the program (reconstructed from the internal tree) are of general interest, as are the options to print the full internal tree, and the entity table (that is the symbol table information).

### A2.12.2 GNAT Crashes

GNAT may experience problems in operation, such as aborting with a segmentation fault or illegal memory access, raising an internal exception, or terminating abnormally. If such problems occur, try the solutions described in this section.

The following strategies are presented in increasing order of difficulty, corresponding to the user's skill level and curiosity about the functioning of the compiler.

1. Run *gcc* with the `-gnatf` and `-gnate` switches. The first switch causes all errors on a given line to be reported. In its absence, only the first error on a line is displayed. The `-gnate` switch causes errors to be displayed as soon as they are encountered, rather than after compilation is terminated. Often this is enough to identify the construct that produced the crash.
2. Run *gcc* with the `-v` (verbose) switch. In this mode, *gcc* produces ongoing information about the progress of the compilation and provides the name of each procedure as code is generated. This switch allows you to find which Ada procedure was being compiled when it encountered a code generation problem.
3. Run *gcc* with the `-gnatdc` switch. This is a GNAT specific switch that does for the frontend what `-v` does for the backend. The system prints the name of each unit, either a compilation unit or nested unit, as it is being analyzed.
4. On systems that have *gdb* available (most Unix systems), start *gdb* directly on the *gnat1* executable. *gnat1* is the front-end of GNAT, and can be run independently (normally it is just called from *gcc*). You can use *gdb* on *gnat1* as you would on a C program (but see section “A2.12.3 Using *gdb* for caveats”). The `where` command is the first line of attack; the variable `lineno` (seen by `print lineno`), used by the second phase of *gnat1* and by the *gcc* backend, indicates the source line at which the execution stopped, and `input_filename` indicates the name of the source file.

### A2.12.3 Using *gdb*

*gdb* awaits modifications to handle Ada properly, and for now can only be used as it would be for a C program. In the meantime, the following naming conventions allow you to find the Ada entities defined in your program:

- The names of all entities (variables, subprograms, and so on) are converted to lowercase letters.
- Entities in library package declarations have the form `package_name__subprogram_name`. Note the double underscores separating package name from subprogram name.

See section “A2.12.3 The Naming Scheme of GNAT” for more information.

Exceptions can be caught by breaking in the `__gnat_raise` function and then entering a `bt` or `where` command.

## A2.13.0 Performance Considerations

The GNAT system provides a number of options that allow a trade off between

- performance of the generated code;
- speed of compilation;
- minimization of dependencies and recompilation;
- and the degree of runtime checking.

The defaults if no options are selected are aimed at improving the speed of compilation and minimizing dependences at the expense of performance of the generated code:

- no optimization
- no inlining of subprogram calls
- all runtime checks enabled except overflow and elaboration checks

These options are suitable for most program development purposes. This chapter documentation describes how you can modify these choices.

### A2.13.1 Controlling Runtime Checks

By default, GNAT produces all runtime checks except arithmetic overflow checking for integer operations (including division by zero) and checks for access before elaboration on subprogram calls.

Two *gnat* switches, `-gnatp` and `-gnato` allow this default to be modified. See section “A2.4.3.3 Runtime Checks.”

Our experience is that the default is suitable for most development purposes.

We treat integer overflow and elaboration checks specially because these are quite expensive and in our experience are not as important as other runtime checks in the development process.

Note that the setting of the switches controls the default setting of the checks. They may be modified using either `pragma Suppress` (to remove checks) or `pragma`

Unsuppress (to add back suppressed checks) in the program source.

## A2.13.2 Optimization Levels

The default is optimization off. This results in the fastest compile times, but GNAT makes absolutely no attempt to optimize, and the generated programs are considerably larger and slower. You can use the '-On' switch, where n is an integer from 0 to 3, on the gcc command to control the optimization level:

- '-O0' no optimization (the default)
- '-O1' medium level optimization
- '-O2' full optimization
- '-O3' full optimization, and also attempt automatic inlining of small subprograms within a unit (see section "A2.13.3 Inlining of Subprograms").

The penalty in compilation time, and the improvement in execution time, both depend on the particular application and the hardware environment. You should experiment to find the best level for your application.

**Note:** Unlike the case with some other compiler systems, gcc has been tested extensively at all optimization levels. There are some bugs which appear only with optimization turned on, but there have also been bugs which show up only in *unoptimized* code. Selecting a lower level of optimization does not improve the reliability of the code generator, which in practice is highly reliable at all optimization levels.



### A2.13.3 Inlining of Subprograms

A call to a subprogram in the current unit is inlined if all the following conditions are met:

- The optimization level is at least `'-O1'`.
- The called subprogram is suitable for inlining: It must be small enough and not contain nested subprograms or anything else that `gcc` cannot support in inlined subprograms.
- The call occurs after the definition of the body of the subprogram.
- Either `pragma Inline` applies to the subprogram or it is very small and automatic inlining (optimization level `'-O3'`) is specified.

Calls to subprograms in `with`'ed units are normally not inlined. To achieve this level of inlining, the following conditions must all be true:

- The optimization level is at least `'-O1'`.
- The called subprogram is suitable for inlining: It must be small enough and not contain nested subprograms or anything else `gcc` cannot support in inlined subprograms.
- The call appears in a body (not in a package spec).
- There is a `pragma Inline` for the subprogram.
- The `'-gnatn'` switch is used in the `gcc` command line

Note that specifying the '-gnatn' switch causes additional compilation dependencies. Consider the following:

```
package R is
  procedure Q;
  pragma Inline Q;
end R;
package body R is
  ...
end R;
with R;
procedure Main is
begin
  ...
  R.Q;
end Main;
```

With the default behavior (no '-gnatn' switch specified), the compilation of the `Main` procedure depends only on its own source, '`main.adb`', and the spec of the package in file '`r.ads`'. This means that editing the body of `R` does not require recompiling `Main`.

On the other hand, the call `R.Q` is not inlined under these circumstances. If the '-gnatn' switch is present when `Main` is compiled, the call will be inlined if the body of `Q` is small enough, but now `Main` depends on the body of `R` in '`r.adb`' as well as the spec. This means that if the body is edited, the main program must be recompiled. Note that this extra dependency occurs whether or not the call is in fact inlined by `gcc`.

**Note:** The `gcc` switch '-fno-inline' can be used to prevent all inlining. This switch overrides all other conditions and ensures that no inlining occurs. The extra dependencies resulting from '-gnatn' will still be active, even if the '-fno-inline' switch is used.

---

## APPENDIX B

---

### Resources: Mach<sup>Ten</sup> Application

The Mach<sup>Ten</sup> application includes several resources that define important configuration values and options. The resources listed below may be viewed and changed with *ResEdit*, to customize Mach<sup>Ten</sup> operation to your circumstances. Changing undocumented Mach<sup>Ten</sup> resources may result in unexpected failures.

#### DMOD

The 'DMOD' resource, ID -8192, specifies the default UNIX access privilege modes for Macintosh folders. The default mode of 777 (Octal) permits read, write, and search permission by everyone. The default access modes assigned may be overridden using the Mach<sup>Ten</sup> *chmod(1)* command. The default Macintosh folder owner and group is fixed by Mach<sup>Ten</sup> at "root" and "wheel", respectively.

**Note:** When modifying the DMOD resource, use the hexadecimal number equivalent of the octal mode bits you wish to set, for example: enter mode "755" as \$1ED.

#### FMOD

The 'FMOD' resource, ID -8192, specifies the default UNIX access privilege modes for Macintosh files. The default mode of 777 (Octal) permits read, write and execute permission by everyone. The default access modes assigned may be overridden using the Mach<sup>Ten</sup> *chmod(1)* command. The default Macintosh file owner and group is fixed by Mach<sup>Ten</sup> at "root" and "wheel", respectively.

**Note:** When modifying the FMOD resource, use the hexadecimal number equivalent of the octal mode bits you wish to set, for example: enter mode "755" as \$1ED.

#### Luid

The Luid resource controls which users are permitted to launch Macintosh applications from within Mach<sup>Ten</sup>. The Luid resource has resource type of 'Luid' and a resource ID of -8192 (or 0xE000).

The Luid resource contains a variable length list of numeric uids. A uid is defined to be an "unsigned short". A ResEdit template for the Luid resource is included in Mach<sup>Ten</sup>.

If the Luid resource does not exist or is empty, only the user *root* may launch Macintosh applications. Otherwise, each entry in the resource is compared to the uid of the current user to check for a match. If a match is found, that user is permitted to launch Macintosh applications. The uid 0xFFFF is a wildcard, which matches all users, allowing any user to launch applications. This is the default for Mach<sup>Ten</sup>.

## ROOT

The 'ROOT' resource, ID -8192, names the FFS file which contains the Mach<sup>Ten</sup> root file system. The name is first looked for in the folder from which Mach<sup>Ten</sup> is launched. The default name is "MachTen\_FFS".

---

## APPENDIX C

---

### Suggested Reading for Programming Languages

The following books may be helpful in understanding and using the indicated programming language.

#### Ada

Ada 95: Problem Solving and Program Design, by Michael Feldman and Elliot Koffman. Addison-Wesley, 1996; ISBN: 0-201-87009-6.

This book is suitable as a CS1 text with Ada as the language of instruction, and the last few chapters, combined with some language-independent algorithm theory, cover the rest of the Ada language in sufficient depth to serve as the language-specific basis of a CS2 course.

Note that Prof. Feldman has made available online the first eight chapters of the Instructor's Manual for the book, available at:  
<http://www.seas.gwu.edu/faculty/mfeldman/cs1-im/index.html>.

Ada 95: The Lovelace Tutorial, by David A. Wheeler  
Springer-Verlag; ISBN: 0-387-94801-5.

This tutorial introduces Ada to those who already know another programming language. It is divided into a large number of small sections, most of which end in a question, to improve understanding.

Ada 95: The Craft of Object-Oriented Programming, by John English  
Prentice Hall, September 1996.

This book introduces Ada-as-a-first-language. The site contains 3 sample chapters, a downloadable set of examples, and related bits and pieces.

Software Engineering with Ada, Third Edition, by Grady Booch, Doug Bryan, Charles Peterson; Addison Wesley.

Ada in Action (with Practical Programming Examples), by Do-While Jones.  
Available on the internet at:  
[ftp://owens.ridgecrest.ca.us/pub/users/d/do\\_while/ada\\_in\\_action](ftp://owens.ridgecrest.ca.us/pub/users/d/do_while/ada_in_action)  
Alternative site:  
[ftp://lo-pan.ridgecrest.ca.us/pub/users/d/do\\_while/ada\\_in\\_action](ftp://lo-pan.ridgecrest.ca.us/pub/users/d/do_while/ada_in_action)

**Books Suitable for a First Course in Programming**, suggested by Michael Feldman, and available at the Ada Home Page, <http://www.adahome.com>. These books are written especially for students without programming experience, who are learning Ada as their first language.

Ada from the Beginning, Second Edition, by J. Skansholm  
Addison-Wesley, 1994; ISBN: 0-201-62448-6.

Introduction to Programming Concepts and Methods with Ada,  
by J.F. Smith and T.S. Frank,  
McGraw-Hill, 1994; ISBN: 0-07-911725-2.

## C

Teach Yourself the Unix C Shell in 14 Days (Unix Library), by D. Ennis/  
J. C. Armstrong, Jr.; Sams, 1994; ISBN:0672305402.

Teach Yourself Advanced C in 21 Days, by Bradley Jones/Gregory Guntle;  
Sams, 1994; ISBN:0672304716.

## C++

The C++ Programming Language, Second Edition, by Bjarne Stroustrup  
Addison Wesley; ISBN: 0-201-53992-6.

Practical C++ Programming, by Steve Oualline  
O'Reilly & Associates, Inc.; ISBN: 1-56592-139-9.

Teach Yourself C++ Programming in 21 Days, by Jesse Liberty

## Objective-C

Object Oriented Programming: An Evolutionary Approach, Brad J. Cox, Andrew J. Novobilski Addison-Wesley, 1991; ISBN: 0-201-54834-8 (Japanese: 4-8101-8046-8).

The first book on Objective-C, which actually is a book on object oriented system development using Objective-C.

Objective-C: Object Oriented Programming Techniques, Lewis J. Pinson, Richard S. Wiener Addison-Wesley, 1991; ISBN 0-201-50828-1 (Japanese: 4-8101-8054-9).

Includes many examples, discusses both Stepstone's and NeXT's versions of Objective-C, and the differences between the two.

An Introduction to Object-Oriented Programming, Timothy Budd, Addison-Wesley; ISBN 0-201-54709-0 (Japanese: 4-8101-8048-4).

An intro to the topic of OOP, as well as a comparison of C++, Objective-C, Smalltalk, and Object Pascal

NeXTSTEP Programming Step ONE: Object-Oriented Applications, Simson L. Garfinkel, Michael K. Mahoney; TELOS/Springer-Verlag, 1993 (tel: (800)SPR-INGE).

It's updated to discuss NeXTSTEP 3.0 features (Project Builder, new development environment) but doesn't discuss 3DKit or DBKit.

NeXTSTEP Object Oriented Programming and the Objective C Language. Addison-Wesley, 1993; ISBN 0-201-63251-9 (Japanese: 4-7952-9636-7).

This is also available on the World Wide Web at:  
<http://www.next.com/Pubs/Documents/OPENSTEP/ObjectiveC/objctoc.htm>.

This book describes the Objective-C language as it is implemented for NeXTSTEP. While clearly targeted at NeXTSTEP, it is a good first-read to get to learn Objective-C.

## Articles

*Why I need Objective-C*, by Christopher Lozinski.  
Journal of Object-Oriented Programming (JOOP), September 1991.  
Contact [info@bpg.com](mailto:info@bpg.com) for a copy and subscription to the BPG newsletter.

This article discusses the differences between C++ and Objective-C in great detail and explains why Objective-C is a better object oriented language.

## Java

Instant Java, by John A. Pew  
SunSoft Publishing

Instant Java is the easy, practical way to add Java applets to your Web pages! The following are among the applets included in the book: Audio, Multiple Simultaneous Animations, Image Maps, Ticker Tapes and more.

Java Manual of Style, by Nathan Gurewich/Ori Gurewich  
Ziff-Davis Press; ISBN: 156276408X.

Includes: The building blocks of Java programming, how to use the Java Developer Tool Kit, and how to create applets that will maximize the interactivity in your site.

The Java Programming Language(The Java Series), by Ken Arnold/  
James Gosling ;Addison Wesley; ISBN: 0201634554.

Co-authored by the creator of Java technology, The Java Programming Language is the definitive resource for all serious Java programmers.



## X11

The following books published by O'Reilly & Associates are recommended for learning more about X:

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*. (*Motif* is not included with Mach<sup>Ten</sup>.)

Volume 4: X Toolkit Intrinsic Programming Manual

Volume 5: X Toolkit Intrinsic Reference Manual

Comprehensive guides to programming with the X library and X 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<sup>Ten</sup> X Window Software package.



---

# INDEX

---

## Symbols

---

\$ prompt 51, 75  
&, using for background processing 85

## A

---

access permissions 89, 92  
accessing

- Mac files from UNIX applications 110
- source files 30

Ada 1, 2, 12, 224

- documentation 12

ADDRINFO\_\* variable 65  
adduser command 75, 193  
administration 63–86

- file
  - major 106
  - system 115
- login accounts 80
- Mach<sup>Ten</sup> 63

AfterStep

- desktop 5
- X Window manager 5, 260

aliases, defined 93  
allocating memory 36  
allocation blocks, defined 91  
Alpha 2  
Apache 173  
Apple

- menu 53
- Remote Access 293

AppleDict 159  
AppleDouble 139  
AppleShare access privileges (NFS) 135  
AppleSingle encapsulation 110

AppleTalk 65, 124, 293  
applications, hybrid 9  
archiving file systems 120  
AutoDoubler, disabling 16  
automatic

- booting of Mach<sup>Ten</sup> 67
- mounting
  - Desktop NFS volumes 132
  - removable media 108

## B

---

background program execution 85  
backing up

- file systems 120
- tar command 121

baud rates 152, 187

- table 187

BBEditLite 2, 12, 173  
Berkeley

- Internet Name Domain (BIND) 170
- r-series commands 185

BINA file types 97, 98  
binary file types 97, 98  
BIND (Berkeley Internet Name Domain) 170  
BLK file types 99  
block devices 93, 99  
booting up Mach<sup>Ten</sup> automatically 67  
Bourne shell 49, 72  
broadcasting messages 81  
buckets, defined 235  
buffers

- increasing the number of 22
- UFS and NFS 22, 37

bulletin board folders 138  
bwnfsd 130

UID and GID mapping 133

## C

### C

compiler 1, 225

shell 49, 72

C++ compiler 1, 226

cable, GPI 188

cabling and connectors for serial ports 186

caching 21

Carnegie Mellon Mach kernel 6

case sensitivity in file names 94

cc1obj 215

cc1plus 215

CFM, *See code fragment manager*

characters

non-printable 95

special keyboard 83

checking file systems 116

chpass command 77

CHR file types 99

chroot command 77

CLEANUP\_tmp variable 66

code fragment 6, 213, 219, 231

loader 220

manager 242

compilers

Ada 1, 224

C 1, 225

C++ 1, 226

engines

cc1 215

cc1obj 215

cc1plus 215

f771 215

gnat1 215

Fortran 1, 227

Java 1, 228

Objective-C 226

compile-time libraries 220

naming conventions 221

compiling source code 224

Ada 224

C 225

C++ 226

Fortran 227

Java 228

Objective-C 226

component separators for file names 95

configuration

DNS lookups 179

example 43

options 21

via Mach<sup>Ten</sup> control panel 33, 65

configuration screens 34

general 35–36

host name 35

time zone 36

memory

maximum number parameters 37

networking 35, 38–43

enable

incoming

connections 39

mail 39

IP forwarding 39

IP addresses 40–42

maximum transmission unit

settings 42

netmasks 42

connections, incoming 39

console window 19

control panels 33–43

date & time 36

file sharing 21

- Mach<sup>Ten</sup> 18, 46, 168, 172
  - configuring 18, 33, 65
  - warning dialog box 32
- map 36
- memory 21, 22
- networking 180, 183
- sharing setup 35
- TCP/IP 15, 38, 177
- X Windows, configuring 296–298
- cooperative sharing 6
- counts, link 90
- cpp 214
- cron 23
  - daemon 47
- cross-development tools and targets 240
- .cshrc file 72, 76
- cu command 189
- cylinder group, defined 88

## D

- daemons 46
  - cron 47
  - getty 24
  - httpd 25, 47, 173
  - inetd 24, 47
  - lockd 132
  - lpd 25, 47
  - lwsrv 149
  - Mach<sup>Ten</sup>, table of 47
  - macmntd 47
  - mactcpd 47
  - mountd 47, 125
  - named 47, 171
  - NFS 24
  - nfsd 47, 125
  - portmap 47, 125, 143, 147
  - rc 47

- routed 24
- sendmail 24, 47
- syslogd 24, 47
- wind 25, 47, 69
- yppbind 143, 147
- data
  - files, Mach<sup>Ten</sup> 48
  - fork 139
- databases
  - etc/rc 46
  - etc/rc.conf 46
- date & time control panel 36
- Debugger 248
- debugging 231
  - Debugger 248
  - environment variables
    - DEBUGGERFIRST 235
    - MEMSTATS 235
    - STACKCHK 235
  - gdb 231
  - Macintosh debuggers 248
  - MacBug 232, 248
  - Metrowerks debugger 233
- default gateway 42
- Desktop NFS 130–141
  - access
    - permissions
      - Make Changes 137
      - See Files 137
      - See Folders 137
    - privileges, strategies for 138
- AppleDouble
  - file support 130
  - storing Mac files 139
- AppleSingle
  - file support 130
  - storing Mac files 139
- authentication options 133

- user ID and password on
    - command line 135
  - username and password
    - on command line 134
    - prompt 134
- comparison with Apple File
  - Sharing 139
- creators and types 141
- credentials of user 133
- defined 130
- DOS extensions 140
- locked folders 138
- Macintosh record locking 132
- DeskWriter baud rates 152
- /dev/ttya 186
- /dev/ttyb 186
- development tools 9, 213–248
  - UNIX 1
  - X Windows 1
- Dext resource 140
- df command 23
- dfork 111
- dfork.text 113
- directory link counts 97
- disabling
  - AutoDoubler 16
  - logins 82
  - virus protection software 16
- disk
  - caching 21
  - drivers 21
  - footprint 8
  - reformatting for performance 15
  - requirements 14
  - space limitations 23
- diskless NFS clients 127
- distributed file system 1, 184
- DNS, *See domain name server*

- documentation
  - Ada 12
  - GNAT 12
  - Mach<sup>Ten</sup> 12
  - Objective-C 12
- domain name 42
  - requirements for installation 15
  - resolver 168, 172
  - server 42
    - BIND, enabling 179
    - boot file 171
    - configuring Mach<sup>Ten</sup> as 170
    - lookups, configuration 179
    - using Mach<sup>Ten</sup> as 170
- domainname 144
- DOS extensions 140
- drop folders 138
- dtmount command 130
- dt nfs program 130
- DTNFS, *see Desktop NFS*
- dual stack configuration 164, 177
- dump command 120, 122
- dynamic
  - linking 7, 219
  - memory configuration 7

## E

---

- edata symbol 245
- edit menu 54–55
- editing tools 114
  - copy & paste 114
  - Macintosh editing applications 114
  - UNIX editors 114
- editors, changing default 70
- electronic mail 180
- emacs 114
- enable

- incoming
  - connections 39
  - mail 39
- IP forwarding 39
- end symbol 245
- environment variables
  - DEBUGGERFIRST 235
  - MEMSTATS 235
  - STACKCHK 235
- error logging, X server 285
- /etc/rc file 23
- /etc/exports file 124
  - example 126
- /etc/fstab file 48
  - example 127
- /etc/group file 78, 79
- /etc/hosts
  - file 48, 168, 170
  - example 171
  - table 168
- /etc/hosts.equiv file 154
- /etc/hosts.lpd file 154
- /etc/localtime file 48
- /etc/master.passwd file 77, 82
- /etc/motd file 48, 82
- /etc/named directory 170
- /etc/named/boot file 171
  - example 172
- /etc/net\_search\_rules files 168
- /etc/nis.conf file 143
- /etc/nis.conf file 144
- /etc/nsswitch.conf file 143, 145
  - example 145
- /etc/passwd file 78
- /etc/printcap file 150, 161
  - modifying 151
- etc/rc database 46
- /etc/rc.conf file 48

- etc/rc.conf database 46
- /etc/rc.local file 48
- /etc/remote database 198
- /etc/resolv.conf file 172, 48
- /etc/sendmail.cf file 48
- /etc/ttys file 48
- /etc/yp.conf file 143
  - modifying 144
- etext symbol 245
- Ethernet 38, 39, 65, 124, 169
- EtherTalk 14
- Eudora 182
- execute permission 89
- exporting via NFS 124
- eXtended Common Object File Format (XCOFF) 219

---

## F

- f771 215
- Fast File First Aid 117
- fast file system 87–93
  - backing up with tar command 121
  - defined 87
  - native 7, 9
  - overview 88
  - within a file 93
- FFS, *See fast file system*
- FFS\_Installer 13
- FIFO file types 100
- file
  - access, memory mapped 8
  - contents 92
  - .cshrc 76
  - /etc/group file 78, 79
  - group 78
  - header 222
  - .login 72, 86

- .profile 72, 86
  - macros 222
  - pre-defined names 222
- linked 96
- locked 97
- Macintosh
  - dfork 111
  - finderinfo 111
  - manipulating 111
  - restool 111
  - rfork 111
- Makefiles 230
- manipulation of text 112
- memory mapped 242
- menu 53
  - X desktop 252
- names
  - case sensitivity 94
  - component separators 95
  - maximum number of
    - characters 94
    - non-printable characters 95
    - size limitations 89, 92
- open, maximum number allowed 37
- password 77
- PEF format 9
- permissions 18, 101
  - example 101
- sharing 3, 21
  - control panel 21
- shell files, example 86
- system
  - access permissions 89
  - administration 115
  - archiving 120
  - backing up 120
  - checking 116
  - creating 115
  - distributed 1
  - fast 87-90
  - Mach<sup>Ten</sup> 87-122
  - Macintosh 91-93
  - native 7, 9
  - organization 88
    - file counts 92
    - folders 91
    - volumes 91
  - overview 91
  - removing 118
  - repairing 116
  - root 103
  - security 135
    - AppleShare access 135
    - UNIX access 136
  - UNIX 94
  - unmounting 118
- types
  - BINA 97
  - binary 98
  - BLK 99
  - CHR 99
  - FIFO 100
  - HLNK 100
  - LINK 99
  - MUMM 97
  - SHLB 100
  - SOCKET 99
  - TEXT 97, 98
- File Transfer Protocol, *See* FTP
- Finder, defined 3
- finderinfo 111
- flex 213
- floppies, formatting 109
- flow control primitives 72
- FolderBolt 73
- folders



- drop 138
- shared 138
- fonts
  - lists for printing 159
  - menu 61
  - X server
    - default path 287
    - network font server 288
    - providing over network 289
    - xfs 289
- foreign file format 183
- formatting floppies 109
- Fortran 1, 227
- fragmentation, memory 23, 245
- fsck command 116
- FTP 168
  - anonymous 183
  - overview 183

---

**G**

---

- g++ 214
- g77 214
- GATEWAY variable 67
- gateway
  - default 42
  - defined 176
- gcc 214
- gdb 214, 231
- General Purpose Interface (GPI) cable 188
- getty 24, 189
- GNAT documentation 12
- gnat1 215
- gnatmake 214
- GNU compiler 2, 224
- GPI (General Purpose Interface) cable 188
- Greenwich Mean Time 36
- group

- cylinder, defined 88
- file 78, 79
- IDs 71, 75
- wheel 72

## H

---

- halt command 70
- hard links 90, 96, 100
- hardware flow control 186, 188, 200
- header
  - compression 202
  - files 222
    - macros 222
    - pre-defined names 222
- heap
  - Mach<sup>Ten</sup> 243
  - size 244
  - system 243
- HFS
  - file system 91-94
  - partition 14
- HLNK file type 100
- home
  - directory 71
  - page, default 173
- host
  - identifier 175
  - name 168
    - defined 35
    - to-internet address mapping 48
    - translating to internet
      - address 170
  - table 170
- HOSTNAME variable 65
- hot keys, X window manager 256
- HTML (Hypertext Markup Language) 173
- HTTP 168

httpd daemon 25, 47, 173

HyperText Markup Language  
(HTML) 173

hypertext transfer protocol daemon  
(httpd) 173

## I

---

IFCONFIG\_\* variable 65

include files 213

incoming connections, enabling 39

indent 213

index.html file 12

inetd daemon 24, 39, 47

init command 46

initialization of Mach<sup>Ten</sup> 46

inode, defined 88

Install.conf file 45

Installation Guide 14–18

installing Mach<sup>Ten</sup> 11–32

international keyboards 259  
    configuring 284

internet

    address 168

        identifier 175

        obtaining 168

    defined 168

    host, using Mach<sup>Ten</sup> as 168

Internet Protocol (IP) 163

IP

    addresses

        AppleTalk 40

        default gateway 42

        domain name server 42

        Ethernet 41

        requirements for installation 15

        TokenRing 41

    unique 40

    datagrams 176

    forwarding 24, 39

        invoking 176

    packets, forwarding 39

    protocol 163

## J

---

Java 1, 228

    Kaffe 2

## K

---

Kaffe 2, 228

keyboard

    Alt keys 285

    characters, special 83

    international 259

        configuring 284

    mapping 259, 285

    X Meta keys 285

kill command 23, 85

killing a program 85

## L

---

launching Mach<sup>Ten</sup> 18, 45

ld (link editor) 214, 229

libraries

    compile-time 220

        naming conventions 221

    Mach<sup>Ten</sup> 220

    run-time 220

        naming conventions 221

    shared 6, 7, 100, 219, 243

        overview 219

        producing 220

    software development 215–217

- Tenon 27
- UNIX 7
  - traditional 221
- X Windows 216–217

- link editor (ld) 229

- LINK file types 99

- linked files 96

- links

- count 90
  - directory counts 97
  - dynamic 7, 8, 219
  - hard 90, 96, 100
  - symbolic 90

- loader 229

- local printing 150

- LocalTalk 14, 38, 39, 169

- lockd daemon 132

- locked

- files 97
  - folders 138

- logging in 49, 71, 74

- SLIP 207

- logical blocks, defined 91

- login

- accounts 71
    - administrative 80
    - home directory 71
    - user environment 71
  - authentication, SLIP 205
  - console 49, 50
  - directory 71
  - disabling 82
  - privileges 72

- .login file 72, 86

- loopback mode 169

- lp program 162

- lpd daemon 25, 47, 150

- lpq program 162

- lpr program 150

- lwsrv

- configuration files 158

- daemon 149

- diagram 158

- enabling 160

## M

---

- Mach kernel 6

- Mach<sup>Ten</sup>

- administration 63–86

- application 11

- booting 18, 45

- automatic 67

- manual 67

- components of 11

- configuration

- automatic startup of portmap and ypbind 148

- options 21

- console window 18, 19

- control panel 18, 33, 168, 172

- configuring 18

- startup files 46, 65

- warning dialog box 32

- daemons 46

- table 47

- defined 1

- desktop 3

- documentation 12

- GNAT documents 217

- HTML 217

- man pages 217

- PDF 217

- domain name service 170

- configuring 170

- file systems 87–122
- folder contents 17
- FTP 183
- heap size 244
- initialization 46
- Installer 11
- installing 11–32
- internet host 168
- launching 18, 45
- libraries 220
- licensing information 11
- login console 49, 50
- menus 52
- NFS 184
- passwords 49, 51
- performance tuning 21
- POP mail service 180
- porting software to 241
- printing 149
- programming environment 213
- quitting 70
- reconfiguration
  - automated 28
  - manual 29
- reinstallation 26
- security 73
- sources, accessing 30
- starting 18, 45
  - automatic 67
  - manual 67
- startup environment 65
- symbol information 230
- system architecture 7, 8
- TCP stack vs.
  - OpenTransport 165–166
- telnet 184
- terminal window 51
  - desktop 4
- troubleshooting 31–32
- user guide, online 11
- utilities 12
- virtual machine 6, 11
- Web service 173
- windows 52
  - tailoring 68
- MachTen Conf file 45
- MachTen Prefs file 45
- MACHTEN variable 69
- /MachTen/misc/rc.config file 48
- Macintosh
  - applications, developing 236
    - construction 239
    - header definition files 236
    - interface libraries 237
    - startup routine 238
  - debuggers 248
  - desktop 3
  - DNS servers and OpenTransport 179
  - file system (HFS) 91–93, 94
    - access permissions 92
    - aliases 93
    - file contents 92
    - file names 92
    - organization 91
    - time stamps 93
    - volumes and folders 91
  - files, manipulating 111
    - dfork 111
    - finderinfo 111
    - restool 111
    - rfork 111
  - foreign file format 183
  - record locking 132
  - toolbox bindings 2
  - volumes
    - mounting 107, 118

- unmounting 108
- macmntd daemon 47
- MacOS 6, 11
  - rebooting 18
- MacPerl 2
- MacsBug 232, 248
- MacTCP applications 179
- mactext 112, 113
- mail
  - daemon 24
  - electronic 180
  - incoming 39, 47
  - program
    - defined 181
    - examples 182
  - sending/receiving 39
- .mailrc file, example 181
- make program 214, 246
- Makefiles 230
- man pages 64
- managing your system 84–86, 119
- manual startup of Mach<sup>Ten</sup> 67
- map control panel 36
- mapping UIDs and GIDs via bwnfsd 133
- maximum transmission unit settings 42
  - SLIP 203
- memory
  - allocating 9, 36, 242–243, 245
  - configuration
    - dynamic 7
    - screen 36
  - control panel 21, 22
  - fragmentation 23, 245
  - limitations 23
  - manager 9
  - mapped files 242
    - access 7
  - page boundary 245
  - physical 22
  - real, programming issues 245–246
  - requirements 7, 8, 14, 243
  - running out during compilation 246
  - sbrk 245
  - stack overrun 241
  - sticky bits 246
- menu bar
  - running X server without 255
  - shortcuts 255
  - X desktop 251
- menus
  - Apple 53
  - Edit 54–55
  - File 53
  - Fonts 61
  - Mach<sup>Ten</sup> 52
  - Order 59–60
  - Positions 57
  - Size 58
  - Window 56
- message of the day 48
  - changing 82
- Metrowerks debugger 233
- mkpef 229
- modem
  - connecting to Mach<sup>Ten</sup> serial port 189
  - control translations 188
- Motif Window Manager 260
- mount command 124
  - example 127
- MOUNT\_REMOTE\_FILES variable 66, 128
- mountd daemon 47, 125
- mounting
  - Desktop NFS volumes 131
  - directories, via NFS 124
  - FFS 117

- file systems 117
- Mach<sup>Ten</sup> CD-ROM 30
- Macintosh volumes 118
- NFS clients 126
- permissions 107
- removable media 108
- mouse
  - button mapping 281
  - Logitech 283
  - three button 281–283
    - configuration 281
    - default keystrokes 282
    - example 283
  - X desktop 267
    - clicking and dragging 267
    - pointer shapes 267
- mt utility 121, 122
- MTU
  - defined 42
  - SLIP 43, 203
  - TokenRing 43
- multi-homing 38, 169
- multi-link, multihoming 164
- multiple users, support for 71
- multitasking, pre-emptive 6
- MUMM file types 97
- mwm 260

## N

---

- name server 171
- named daemon 47, 171
- NAMED\_BOOT variable 67
- native fast file system 7, 9
- net\_search\_rules command 168
- NETMASK\_\* variable 65
- netmasks 42
- network configuration screen 38

- configuration
  - screen 38
- Network File System Protocol
  - Specification (RFC 1094) 125
- Network File System, *see* NFS
- Network Information Service, *see* NIS
- Network Solutions, how to contact 168
- networking 163–211
  - configuration
    - example 43
    - multiple interfaces 169
    - screen 38
  - control panel 180, 183
  - overview 163
- NeXT 5
- NFS 123–141
  - authentication options 133
    - user credentials 133
  - bwnfsd UID and GID mapping 133
  - client
    - defined 123
    - diskless 127
    - mounting 126
    - setting up 125
  - daemons
    - mountd 125
    - nfsd 125
    - portmap 125
    - starting 126
  - defined 123
  - exporting directories 124
  - how it works 124
  - mounting 124
    - volumes on the desktop 130–132
      - automatic 132
  - overview 184
  - record locking 132
  - server

- defined 123
- exporting 126
- setting up 125
- setting up, instructions 128–129
- stateless mode 124
- unmounting volumes on the
  - desktop 131
- using 124
- nfsd daemon 47, 125
- NIS 143–148
  - configuring 143–148
    - testing 147
  - database search order 145
  - defined 143
  - domainname 144
  - nis 145
  - nis+ 145
  - nisplus 145
  - nisserver 144
  - portmap, starting automatically 148
  - START\_portmap 148
  - START\_ybind 148
  - testing 147
  - YP 143
    - defined 143
  - ybind, starting automatically 148
  - ypcat application 147
  - ypmatch application 147
  - ypserver 144
- nis+ 145
- nisplus 145
- nisserver 144
- NIST, defined 143
- nm development tool 214
- non-printable characters 95

---

## O

- Objective-C 226
- olvwm 256, 260
- OpenGL 13
- OpenLook
  - environment, illustrated 266
  - Virtual Window Manager 260
  - window menu, illustrated 274
- OpenTransport
  - configuring 178
  - Macintosh DNS servers 179
  - networking with 15
  - replacing 164
  - using with Mach<sup>Ten</sup> TCP stack 164, 177
  - vs. Mach<sup>Ten</sup> TCP stack 165–166
- operating system, co-resident 3
- optimizing your system 22–25
- order sub-menu 59–60

---

## P

- parameter passing 72
- partitioning your disk 14
- passwd
  - command 51
  - program, invoking 49
- password
  - changing 77, 82
  - file 77
  - master file 77
  - root 74
- pcnfsd 130
- PEF 7, 9, 219, 229, 231
  - code fragments 231
- performance
  - optimizing 21–22, 34
  - tuning 21

- X server 286
- Perl 2, 13
- permissions
  - access 92
  - execute 89
  - file 18, 101
    - example 101
  - mounting 107
  - read 89
  - write 89
- Pine4 182
- pipes 100
- pmake 214, 230
- POP mail server, using Mach<sup>Ten</sup> as 180
- POP3 (Post Office Protocol) 180
- porting software 241
- portmap
  - daemon 47, 125, 143, 147
  - starting automatically 148
- positions sub-menu 57
- POSIX 6
- Post Office Protocol (POP3) 180
- PostScript macros for printing 159
- Power Macintosh 11
  - memory manager 9
- PowerPC
  - binary format 2
  - compiler suite 1
  - development environment 1
  - Executable Format (PEF) 7, 219
- PPP 38, 124, 169, 293
  - multiple clients 196
  - overview 190
  - setting up 190
    - client 194–195
    - server 191–193
- pppd program 193
- pr command 151
- pre-emptive multitasking 6
- PRESERVE\_EDITOR\_FILES variable 66
- printcap program 150
- printenv command 69
- printing 149–162
  - AppleDict 159
  - baud rates 152
  - DeskWriter 151
  - font lists 159
  - ImageWriter 149, 151
  - job status notification 161
  - LaserWriters 149
  - local 150
  - lp management program 162
  - lpd daemon 47, 150
  - lpq program 162
  - lpr program 150
  - lwsrv
    - configuration files 158
    - diagram 158
    - enabling 160
  - overview 149
  - PostScript 149
    - macros 159
  - pr command 151
  - printcap 150
  - PRINTER shell environment
    - variable 156
  - ProcSet 159
  - receiving remote print jobs 155
  - remote spooling 1, 149, 154
    - configuring 154
      - Macintosh applications 157
  - selecting an alternate printer 156
- sharing 3
- spooler database 150
- spooling 11, 149, 149–155



- defined 149
- to remote UNIX printer 154
- status of jobs 162
- text files to PostScript LaserWriter on AppleTalk 151

process IDs 85

ProcSet 159

.profile file 72, 86

program execution, background 85

programming 213–248

- building executable binary image 248
- environment 213
- example 247
- notes 303
- real memory issues 245–246
  - edata symbol 245
  - end symbol 245
  - etext symbol 245
  - fragmentation 23, 245
  - page boundary 245
  - running out during compile 246
  - sbrk 245
  - stack overrun 241
  - sticky bits 246
- source code 218

ps command 23, 84

purging files 23

## Q

---

quitting

- Mach<sup>Ten</sup> 70
- X Windows environment 277

## R

---

ranlib 214

rc daemon 47

RCS (Revision Control System) 213

read permission 89

reboot command 70

rebooting MacOS 18

reconfiguring Mach<sup>Ten</sup>

- automated 28
- manual 29

record locking on Desktop NFS

- volumes 132

reinstalling Mach<sup>Ten</sup> 26–27

remote

- print jobs, receiving 155
- printer spooling 1, 149, 154
  - configuring 154
  - to UNIX printer 154

Remote Procedure Calls (RPC) 124

removing

- FFS 118
- file systems 118
- users 82

repairing file systems 116

requirements

- disk space 14
- memory 7, 8, 14, 36, 243
- system 11, 14

resource

- fork 139
- sharing, example 163
- X Windows 260

restool 111, 214

restore command 120, 122

Revision Control System (RCS) 213

RFC 1094 “Network File System Protocol Specification” 125

rfork 111

root

- directory tree 103
- file system layout 103
- password 74

route command 175, 176  
routed daemon 24

routing  
    IP packets 39  
    LocalTalk and Ethernet 39  
RPC (Remote Procedure Calls) 124  
RS232 cable 200  
r-series commands, overview 185  
run-time libraries 220  
    naming conventions 221

## S

sbrk 245  
scheduling priority slide bar 21, 33–34  
SCSI tape devices 120  
security 73  
    file system 135  
sendmail 24, 48, 180  
    daemon 47  
Serial DMA 187  
serial line  
    communications 186  
        accessing serial ports 186  
        cabling and connectors 186  
    networks 14  
    running X over 293  
serial ports  
    accessing 186  
    cabling and connectors for 186  
    set up, SLIP 203  
servers, NFS 123  
setstackspace 214  
shared  
    folders 138  
    libraries 6, 7, 100, 219, 243  
        overview 219

    producing 220  
sharing  
    control panel 35  
    cooperative 6  
    files 3, 21  
    printers 3  
shells  
    Bourne 49, 72  
    C 49, 72  
    files, example 86  
    UNIX 72  
SHLB file type 100  
Simple Mail Transfer Protocol  
    (SMTP) 168, 180  
size sub-menu 58  
SLIP 38, 65, 124, 169, 293  
    C-shell scripts  
        bringing down, example 211  
        expediting connections 210  
        setting up, example 210  
    default route 202  
    hardware flow control 200  
    header compression 202  
    logging in 207  
    manual connections 206  
    MTU (Maximum Transmission  
        Unit) 43, 203  
    overview 197  
    setting up  
        dialing in 203  
        dialing out 197  
            login authentication 204  
            serial port 203  
            sliplogin program 205  
    starting 207  
    stopping 205, 209  
    testing 208  
    tip

- activity log 198
- configuration database,
  - example 198–200
  - login script 201
- sliplogin program 205
- SMTP (Simple Mail Transfer Protocol) 168, 180
- SOCK file types 99
- sockets 99, 164
- software development 213–248
  - compiler engines
    - cc1 215
    - cc1obj 215
    - cc1plus 215
    - f771 215
    - gnat1 215
  - compiling source code 224
    - C 225
    - C++ 226
    - Fortran 227
    - Java 228
    - Objective-C 226
  - cross-development tools and targets 240
  - documentation 217
  - include files 213
  - libraries 215–217
  - Mach<sup>Ten</sup> default environment 240
  - Macintosh applications 236
  - notes 303
  - overview 213
  - source code 218
  - tools 213–217
    - ar 214
    - as 214
    - building libraries and programs 214
    - cpp 214
    - flex 213
    - g++ 214
    - g77 214
    - gcc 214
    - gdb 214
    - gnatmake 214
    - indent 213
    - ld 214
    - make 214
    - nm 214
    - pmake 214
    - ranlib 214
    - restool 214
    - setstackspace 214
    - yacc 213
  - under X 301
  - X Windows programs 216–217
- source
  - code 218
    - compiling 224
    - files, accessing 30
  - Source\_FFS 13, 30, 218
  - space management 119
  - special keyboard characters 83
  - spooling, printer 1, 11, 149
  - stack overrun 241
  - stamps, time 93, 102
  - START\_cron variable 66
  - START\_inetd variable 66
  - START\_lpd variable 66
  - START\_macmntd variable 66
  - START\_named variable 66, 172
  - START\_portmap 148
  - START\_sendmail variable 66
  - START\_syslogd variable 66
  - START\_yplibind 148
- starting
  - Mach<sup>Ten</sup> 45

- automatically 67
  - manually 67
  - SLIP 207
  - startup environment 65
  - Startup Items folder 67
  - sticky bits 246
  - string substitution 72
  - su command 72, 73, 74
  - superuser 72
  - symbol information 230
  - symbolic links 90
  - syslogd daemon 24, 47
  - system
    - administration files 106
    - architecture 7, 8
    - limitations 25
    - logger daemon 24
    - optimization guidelines 22–25
    - requirements
      - disk space 14
      - memory 7, 14, 36, 243
      - system 11, 14
  - System Folder, protected 15
- T**
- 
- Tab Window Manager 260
  - tailoring
    - Mach<sup>Ten</sup> windows 68
    - UNIX environment 69
  - tape
    - devices 120
      - SCSI 120
    - drives, support for 120
  - tar command 120, 121
    - example 122
  - tcl/tk 2
  - TCP
    - applications 164
    - control panel 15, 38, 177
    - protocol 163
    - stack
      - dual 164, 177
      - Mach<sup>Ten</sup> and OpenTransport 15, 164, 177
  - telnet, overview 184
  - Tenon
    - libraries 27
    - TCP stack 164, 177
  - TERM variable 69
  - TERMCAP variable 69
  - terminal
    - connecting 189
    - emulation, VT100 69
    - supporting 189
    - windows 51
      - changing the name of 68
      - multiple 4
      - UNIX 4
      - xterm 260
  - text file manipulation 112
  - TEXT file types 98
  - three button mouse 281–283
    - configuration 281
    - default keystrokes 282
    - example 283
  - time
    - stamps 89, 93, 102
    - zones 36, 48
  - tip
    - activity log 198
    - command 189, 191, 194
    - configuration database 198–200
    - login script 201
  - TokenRing 38, 41, 65, 124, 169
    - MTU 43

TokenTalk 14  
touch command 82  
Transmission Control Protocol (TCP) 163  
troubleshooting Mach<sup>Ten</sup> 31–32  
TTY variable 69  
twm 260

## U

---

UDP (User Datagram Protocol) 124  
UFS, *See* UNIX file system  
umask 137  
umount command 124  
UNIX  
    access  
        permissions  
            Read 137  
            Search 137  
            Write 137  
        privileges (NFS) 136  
            mapping AppleShare access  
                to UNIX access 137  
    API 6  
    applications 1  
    Berkeley 6  
    command interpreter 49, 72  
    development tools 1  
    editors 114  
    environment  
        example 69  
        managing 84  
        tailoring 69  
    file system 94  
        defined 94  
        file names 94  
            case sensitive file names 94  
            component separators 95  
            maximum number of

                characters 94  
                non-printable characters 95  
    libraries 7, 11  
        traditional 221  
    processes, maximum number  
        allowed 37  
    shell 49  
    terminal windows 4  
    virtual machine 6  
Unix <-> Text 113  
unixtext 113  
unmounting  
    automatic 109  
    directories, via NFS 124  
    file systems 118  
    Macintosh volumes 109  
    NFS volumes 131  
user  
    accounts, setting up 75  
    environment 71  
    IDs 77  
    removing 82  
User Datagram Protocol (UDP) 124  
usr directory tree 104  
uucp command 189

## V

---

var directory tree 105  
/var/log/aculog file 198  
/var/preserve file 23  
variables  
    ADDRINFO\_\* 65  
    CLEANUP\_tmp 66  
    DEBUGGERFIRST 235  
    GATEWAY 67  
    HOSTNAME 65  
    IFCONFIG\_\* 65

MACHTEN 69  
 MEMSTATS 235  
 MOUNT\_REMOTE\_FILES 66, 128  
 NAMED\_BOOT 67  
 NETMASK\_\* 65  
 PRESERVE\_EDITOR\_FILES 66  
 PRINTER 156  
 STACKCHK 235  
 START\_cron 66  
 START\_inetd 66  
 START\_lpd 66  
 START\_macmtd 66  
 START\_named 66, 172  
 START\_sendmail 66  
 START\_syslogd 66  
 TERM 69  
 TERMCAP 69  
 TTY 69  
 VPATH 218, 246  
 XENVIRONMENT 264  
 virus protection software, disabling 16  
 volume, defined 91  
 VPATH variable 218, 246  
 VT100 emulation 69

## W

---

wall command 81  
 Web server, using Mach<sup>Ten</sup> as 173  
 wheel group 72  
 wind daemon 25, 47, 49, 69  
 windows
 

- changing to and from an
  - icon 272–273
- environment 49, 52
- Mach<sup>Ten</sup> 52
  - tailoring 68
- menu 56

displaying 274  
 functions 274–275

X desktop 253  
 terminal 51  
 X desktop 267
 

- moving 268–269
- raising 276–277
- sizing 270–271

.windrc file
 

- X Windows environment 255
  - creating 68
  - defined 68

World Wide Web (WWW) 25, 47, 173

write permission 89

writing to all users 81

WWW (World Wide Web) 25, 47, 173

## X

---

X desktop 249
 

- changing a window to and from an
  - icon 272–273
- file menu 252
- illustrated 250
- menu bar 251
- mouse
  - clicking and dragging 267
  - pointer shapes 267
- moving a window 268–269
- raising a window 276–277
- selecting a window 267
- sizing a window 270–271
- using the mouse 267
- window menu 253
  - displaying 274
  - functions 253–276
  - functions summary 275

- X Display Management, *See* X server, XDM
- X Intrinsic Toolkit 262
- X server
  - automatic launching 255
  - command line options 278–281
  - database 263
  - default font path 287
  - error logging 285
  - getting started 299
  - menu bar shortcuts 255
  - network font server 288
  - performance tuning 286
  - providing fonts over network 289
  - quitting 255
  - running without a menu bar 255
  - starting 250
  - XDM 289–292
    - configuring 290–292
    - session 292
- X Windows 249–303
  - AfterStep Window Manager 260
  - applications 1
    - building 299–300
    - running 301
  - clients
    - starting 257
    - startup script 257
      - example 258–259
  - configuring 257
  - control panels 296–298
  - defined 249
  - desktop 5, 250
  - development
    - environment 301
    - tools 1
  - keyboards 259
    - Alt keys 285
    - international 284
    - X Meta keys 285
  - libraries 216–217
  - manager 5, 260
    - AfterStep 5
    - hot keys 256
    - overview 266
    - starting 266
  - Motif Window Manger 260
  - mouse
    - button mapping 281
    - Logitech 283
    - three button 281–283
      - configuration 281
      - default keystrokes 282
      - example 283
  - mwm 260
  - olwmm 260
  - OpenLook
    - environment, illustrated 266
    - Virtual Window Manager 260
  - over serial lines 293
  - overview 294–296
  - preferences 260
  - quitting 277
  - resources 259, 260
    - administering 265
    - application level 262
    - class names 261
    - command line 264
    - files 261
      - locations 262
    - host system level 264
    - instance names 261
    - syntax 261
    - user level 263
    - X server level 263
  - screen colors 259

- server 1
- Tab Window Manager 260
- twm 260
- X desktop 249, 250
- X Intrinsic Toolkit 262
- XENVIRONMENT variable 264
- xfs font server 289
- .xinit 257
- .Xmodmap file 259
- xprop utility 261
- xterm 260
- XCOFF (eXtended Common Object File  
Format) 219
- xfs font server 289
- .xinit program 257
- .Xmodmap file 259
- xprop utility 261
- .xtmenu program 255

## Y

---

- yacc 213
- YP 143–148
- ypbind
  - daemon 143, 147
  - starting automatically 148
- ypcat application 147
- ypmatch application 147
- ypserver 144