

Heurikon UNIX System V & V.2
Reference Guide

HEURIKON
CORP.

Microcomputers For Industry

Heurikon UNIX - Reference Guide

System V and V.2

Heurikon Corporation
3201 Latham Drive
Madison, WI 53713

(608)-271-8700

Apr 1987

Rev D

The information in this guide has been checked and is believed to be accurate and reliable. HOWEVER, NO RESPONSIBILITY IS ASSUMED BY HEURIKON FOR ITS USE OR FOR ANY INACCURACIES. Specifications are subject to change without notice. HEURIKON DOES NOT ASSUME ANY LIABILITY ARISING OUT OF USE OR OTHER APPLICATION OF ANY PRODUCT, CIRCUIT OR PROGRAM DESCRIBED HEREIN. This document does not convey any license under Heurikon's patents or the rights of others.

Heurikon UNIX - Reference Guide
Heurikon Corporation
Madison, WI

1.	BOOTING THE SYSTEM.....	2
1.1	Winchester Booting.....	3
1.2	Floppy Booting.....	12
1.3	Initial System Configuration.....	13
2.	POWER-DOWN PROCEDURE.....	14
3.	IN CASE OF TROUBLE.....	15
3.1	Cannot Boot.....	15
3.2	Programs that Don't Work.....	15
3.3	Terminal or System Lockups.....	17
3.4	System Crashes.....	18
3.5	Other Problems.....	18
3.6	Reporting Bugs.....	20
4.	THE UNIX MANUALS.....	21
4.1	User's Manual.....	21
4.2	Administrator's Manual.....	25
4.3	User Guide.....	25
4.4	Programming Guide.....	25
4.5	Support Tools Guide.....	26
4.6	Document Processing Guide.....	26
4.7	Administrator Guide.....	26
5.	THE UNIX FILE SYSTEM.....	27
5.1	Structure, File and Directory Names.....	27
5.2	Creating Directories and Files.....	28
5.3	Typical Organization.....	29
5.4	Owners and Permissions.....	30
5.5	Repairing a Damaged File System.....	30
6.	TOUR OF IMPORTANT FILES.....	31
6.1	/etc/passwd.....	31
6.2	/etc/inittab.....	32
6.3	/etc/gettydefs.....	33
6.4	/etc/rc.....	34
6.5	C-Shell Login Scripts.....	35
6.6	Bourne-Shell Login Scripts.....	37
6.7	/etc/termcap and terminfo.....	38
6.8	The Clock Daemon.....	38
6.9	News.....	40
6.10	/etc/motd.....	41
6.11	/dev.....	41
7.	THE SHELL.....	42

7.1	Csh Alias Feature.....	42
7.2	Csh History Feature.....	42
7.3	Wild Cards and Expansions.....	43
7.4	Shell Scripts.....	43
7.5	Special Characters.....	46
7.6	Redirection of I/O and Pipes.....	47
7.7	Background Commands.....	49
7.8	Use of Shell Variables.....	50
7.9	Shell Layering.....	51
8.	INTERESTING COMMANDS.....	52
8.1	Process Status.....	52
8.2	Change Directory and List.....	52
8.3	Change Directory and Print Working Directory.....	52
8.4	Pattern Search.....	52
8.5	Display File Contents.....	52
8.6	Display Who is Logged On.....	53
8.7	Sleep.....	53
8.8	Display Environment and Shell Variables.....	53
8.9	Date and Time.....	53
8.10	Translate Characters.....	53
8.11	Copy, Move and Remove.....	53
8.12	Display File Types.....	53
8.13	Check Spelling.....	53
8.14	Echo Arguments.....	54
8.15	Time a Command.....	54
8.16	Send Mail.....	54
8.17	Display Terminal Options.....	54
8.18	Repeat the Previous Command.....	54
8.19	Display History and Aliases.....	54
8.20	Correcting Simple Errors.....	54
8.21	Repeating a Command.....	54
8.22	Head and Tail.....	54
9.	OTHER THINGS TO LEARN (TOOLS).....	55
9.1	Summary.....	55
9.2	Move vs. Copy vs Link.....	56
9.3	The Visual Editor.....	57
9.4	Other UNIX Editors.....	59
9.5	Compiling a 'C' Program.....	60
9.6	Linking C, FORTRAN, Pascal and Assembler Programs.....	60
9.7	The Make Command.....	62
9.8	nroff/troff.....	64
9.9	awk.....	67
9.10	SCCS.....	67
9.11	Debuggers.....	67
10.	ADMINISTRATIVE FUNCTIONS.....	69
10.1	Adding New Users - Removing Old Users.....	69
10.2	Managing Processes.....	72
10.3	Setting /etc/motd.....	72

10.4	Monitoring File System Usage.....	72
10.5	Garbage Collection.....	73
10.6	Examining Log Files.....	74
10.7	Setting the Date and TZ.....	74
10.8	Checking the Nodename.....	76
10.9	Running Vchk.....	76
10.10	Init.....	77
10.11	Adjusting /etc/inittab and /etc/rc.....	78
10.12	Mail Aliases (System V.0).....	78
10.13	System Security.....	79
10.14	System Backups.....	80
10.15	Other Things to Watch.....	80
11.	USING FLOPPY DISKETTES AND TAPE.....	81
11.1	Floppy Disk.....	81
11.2	Streamer Tape.....	85
11.3	Reel-to-Reel Tape.....	89
11.4	Media Interchange.....	90
11.5	Backups via Ethernet.....	91
11.6	Method Comparison.....	91
12.	REBUILDING THE UNIX SYSTEM.....	92
12.1	Floppy or Tape Rebuilding.....	94
12.2	Creating Boot and Rebuild Diskettes.....	98
12.3	Creating Streamer Tape Dumps.....	99
12.4	Writing the Standalone Boot to Winchester.....	100
12.5	Bad Block Checking.....	100
12.6	Manual Disk Format/Mkfs.....	100
12.7	Changing the Swap Space Size.....	101
12.8	File System Check, fsck.....	102
12.9	Creative Use of the Rebuild Diskette.....	103
12.10	Winchester Partitioning.....	104
12.11	Multiple Winchester Drives.....	106
13.	ACCESSING I/O DEVICES (DEVICE DRIVERS).....	107
13.1	phys(2) System Call.....	107
13.2	/dev/mem and /dev/kmem.....	109
13.3	Device Drivers - Reconfiguration Rights.....	110
13.4	Installing a New Device Driver.....	113
13.5	Removing a Device Driver.....	115
13.6	Hints for Writing a New Device Driver.....	116
13.7	Common Device Driver Problems.....	117
13.8	Kernel Routines and Macros.....	117
13.9	Kernel Tables.....	123
14.	I/O ERROR CODES.....	124
14.1	Winchester Errors.....	124
14.2	Reel-to-Reel Tape Errors (M10).....	125
14.3	Floppy Disk Errors.....	126
14.4	Streamer Tape Errors.....	126

15.	MISCELLANEOUS OTHER INFORMATION.....	127
15.1	Floating Point Support.....	127
15.2	SUDH LEDs.....	127
15.3	User Jumpers and LEDs (M10).....	128
15.4	Using Environment Variables.....	129
15.5	/etc/update.....	129
15.6	Sticky Bits and Shared Text.....	129
15.7	Signals.....	130
15.8	RAM Considerations.....	130
15.9	Memory Map.....	131
15.10	Interrupt Usage.....	134
15.11	DMAC Channel Assignments.....	134
15.12	Making pROMs.....	135
15.13	System V.2 Porting to/from System V.0.....	136
15.14	Shared Memory, Semaphore and Messages.....	137
15.15	Accessing Kernel Variables.....	140
15.16	System V.2 Notes.....	142
16.	SERIAL PORT CONFIGURATION.....	144
16.1	Standard Ports.....	144
16.2	Modem Ports.....	144
16.3	Network Ports.....	149
16.4	RS-232-C Connections.....	150
16.5	Device File Setup.....	153
16.6	Sample System Configuration.....	155
16.7	Changing Serial Baud Rates.....	158
16.8	The UUCP System - Hints.....	160
16.9	^Cu Usage.....	164
16.10	The LP Spooler Logic.....	165
16.11	Stty Options.....	170
17.	DEVICE NUMBERING AND NAMING CONVENTIONS.....	172
17.1	Device Numbers and Types.....	172
17.2	Major Device Numbers.....	172
17.3	Minor Device Numbers.....	174
17.4	Device Naming Conventions.....	176
18.	REFERENCE MATERIALS.....	180
19.	APPENDIX A - Changing HK68 Serial Baud Rates.....	181
19.1	Background.....	181
19.2	M10 and V10 Baud Rates.....	181
19.3	Changing the Hbug Configuration Word (M10, V10).....	181
19.4	SBX-SCC Expansion Module Configuration.....	182
19.5	V20 and M220 Default Baud Rates.....	182
20.	APPENDIX B - Sed, Awk Usage Examples.....	183
21.	APPENDIX C - Other Information.....	187
21.1	Additional Documentation.....	187
21.2	Unsupported Commands.....	188

21.3	System Configuration Summary.....	189
22.	READER COMMENT FORM.....	191
23.	COMMAND PAGES.....	193
24.	INDEX.....	215

LIST OF FIGURES

Figure 1.	UNIX Startup Sequence - Flowchart.....	6
Figure 2.	Permuted Index.....	21
Figure 3.	UNIX Documentation.....	21
Figure 4.	Typical UNIX Directory Structure.....	29
Figure 5.	/etc/passwd file (typical).....	31
Figure 6.	/etc/inittab file (typical).....	32
Figure 7.	/etc/gettydefs file (typical).....	33
Figure 8.	/etc/rc file (typical).....	34
Figure 9.	/etc/cshrc file (typical).....	35
Figure 10.	.cshrc file (typical).....	36
Figure 11.	.login file (typical).....	36
Figure 12.	/etc/profile file (typical).....	37
Figure 13.	.profile file (typical).....	37
Figure 14.	/etc/termcap file - portion.....	38
Figure 15.	crontab file (typical).....	38
Figure 16.	/usr/adm/cronlog file (typical).....	39
Figure 17.	calendar file (typical).....	39
Figure 18.	/bin/calendar ~Day enhancement.....	40
Figure 19.	/dev directory (typical).....	41
Figure 20.	~whereis~ script.....	44
Figure 21.	~vmail~ script.....	45
Figure 22.	Linking Languages - C fragment (c.c).....	61
Figure 23.	Linking Languages - FORTRAN fragment (fort.f).....	61

Figure 24.	Linking Languages - Assembler fragment (ass.s).....	61
Figure 25.	Sample makefile (C programs).....	63
Figure 26.	Sample makefile (nroff files).....	63
Figure 27.	~under~ program.....	66
Figure 28.	Adduser script.....	72
Figure 29.	log files.....	74
Figure 30.	/etc/chgnod change.....	76
Figure 31.	Script for Incremental Backup	87
Figure 32.	Rebuild Media Diagram - Typical.....	93
Figure 33.	Winchester Partitions.....	105
Figure 34.	Multiple Drive Configuration - Typical.....	106
Figure 35.	~phys()~ system call.....	107
Figure 36.	~jumper.c~ program (M10, V10).....	108
Figure 37.	Device Driver and Kernel Hooks (Summary).....	113
Figure 38.	System V Physical Memory Map (M10, V10).....	132
Figure 39.	System V.2 Physical Memory Map (V20).....	133
Figure 40.	Semaphore and Shared Memory Example Program.....	140
Figure 41.	nlist(3) example.....	141
Figure 42.	Serial Port Minor Device Format.....	155
Figure 43.	Sample Network Configuration.....	156
Figure 44.	/usr/lib/uucp/L.sys file.....	163
Figure 45.	/usr/lib/uucp/L-devices file.....	163
Figure 46.	/usr/lib/uucp/USERFILE file.....	163
Figure 47.	/usr/spool/lp directory.....	168
Figure 48.	Sample LP Interface Program.....	169
Figure 49.	Example Termio(7) Ioctl Calls.....	171

Figure 50. Guide and Index Preparation Flowchart.....	184
Figure 51. Nroff INDEX script - part 1.....	185
Figure 52. Nroff INDEX script - part 2.....	186

LIST OF TABLES

Table 1. Quick Summary - Boot Procedure.....	3
Table 2. Initial System Configuration.....	13
Table 3. file name forms.....	27
Table 4. Special Characters - partial list.....	46
Table 5. ^vi^ command summary.....	57
Table 6. Floppy Diskette Capacities (Blocks).....	84
Table 7. Reconfiguration Rights - Contents (partial).....	111
Table 8. Kernel Routines and Macros (part 1).....	118
Table 9. Kernel Routines and Macros (part 2).....	119
Table 10. Kernel Routines and Macros (part 3).....	120
Table 11. Kernel Routines and Macros (part 4).....	121
Table 12. Kernel Routines and Macros (part 5).....	122
Table 13. SCSI I/O Errors.....	124
Table 14. Reel-to-Reel Tape Errors, Ciprico Tapemaster (M10 only).....	125
Table 15. Floppy Errors (SBX-FDIO only).....	126
Table 16. HK68 Status LEDs.....	127
Table 17. Signals.....	130
Table 18. DMAC Channel Assignments.....	134
Table 19. Ven-Tel 212-4 Modem Switch Settings.....	146
Table 20. Hayes Smartmodem 1200 Switch Settings.....	147
Table 21. Hayes Smartmodem 1200 Wiring.....	147
Table 22. US Robotics Password Modem Switch Settings.....	147

Table 23.	US Robotics Password Modem Wiring.....	147
Table 24.	Novation Modem Switch Settings.....	148
Table 25.	Novation Modem Wiring.....	148
Table 26.	Connection to a Data Terminal Device.....	150
Table 27.	Connection to a Data Set (Modem).....	150
Table 28.	HK68 RS-232-C Interface.....	151
Table 29.	CDC Eight Channel Expansion RS-232-C Interface.....	152
Table 30.	Serial Port Connections.....	153
Table 31.	/dev/tty nodes (M10 Example).....	154
Table 32.	Sample Network Configuration.....	156
Table 33.	^stty^ options.....	170
Table 34.	Major Block Device Numbers.....	173
Table 35.	Major Character Device Numbers (RAW).....	173
Table 36.	Serial Port Device Assignments (on-card).....	174
Table 37.	SBX-FDIO Minor Device Assignments (M10).....	174
Table 38.	OMTI 5400 - Winchester Minor Device Numbers (M10, V10, M220).....	175
Table 39.	OMTI 5400 - Floppy Minor Device Numbers (M10, V10, M220).....	175
Table 40.	OMTI 5400 - Floppy Type Values (M10, V10, M220).....	175
Table 41.	OMTI 5400 - Streamer Minor Device Numbers (M10, V10, M220).....	175
Table 42.	mknod Device Summary - Partial.....	178
Table 43.	Device Naming Conventions.....	179
Table 44.	Hbug Configuration Word Values.....	182
Table 45.	Hbug Configuration Word Detail.....	182
Table 46.	System Configuration Summary.....	189

Welcome to UNIX.

If you are a first time user, there are many interesting things ahead to learn. In fact, whether you're a beginner or an expert, you will always be learning new things about UNIX. Don't let the apparent complexity scare you; one key to understanding UNIX is to realize that you don't have to learn everything. You only need to learn a few basic commands and know where to look for additional information about new or unfamiliar areas.

This Guide is written for the novice, but it is also useful for the experienced user, because it describes the specific characteristics of our implementation. Use the table of contents and the index to find particular areas of interest. If you are a beginner, be sure to read "UNIX for Beginners" in the UNIX User Guide.

This guide is only a supplement to the UNIX manuals. Our objectives are to:

- ♣ Document Heurikon specific features and procedures. In particular, the methods used to load, backup and rebuild the UNIX system are detailed. We've included information about how to configure the serial ports for connection to a modem, a printer or another processor.
- ♣ Document general UNIX features for the beginner. As an example, we discuss the purpose and formats of some of the system configuration files. These files allow you to alter the operation of UNIX to fit your own personal tastes. We have also included a detailed description of the UNIX manual set, so you will know how to make the most use of them.
- ♣ Document administrative functions. There should always be one person assigned to a system who is responsible for its proper operation. That person, the "system administrator", adds new users, modifies features that effect all users, does file backups and generally assures that the system is in good shape. That job usually requires a few minutes each day to browse through the system, looking for irregularities.

This guide covers UNIX software topics and a few hardware issues. For specifics on particular hardware characteristics, refer to the appropriate hardware manuals.

1 HK68 and Hbug are trademarks of Heurikon Corporation.

2 UNIX is a trademark of AT&T Bell Laboratories.

3 This document was prepared using the UNIX nroff facility and the PWB/mm macros.

1. BOOTING THE SYSTEM

This section describes the procedure to use to boot and start the UNIX operating system. We'd really like you to skim all the manuals and documentation before you start using the system, but we are realists. That's why this section comes first.

The standard method of booting is directly from the Winchester drive. When Heurikon ships a system, the Winchester already contains the bootstrap loader and the complete UNIX system.

In this guide, operator inputs are underscored and/or enclosed in single quotes (e.g., 'bw') and must be followed by a carriage return. For clarity, the act of entering a carriage return is not explicitly stated on most steps; but, you must do so in order for your command to be recognized.

If you make a typing mistake while entering a UNIX command, you can use the backspace key (or the Control-H key) to backup one character. To erase the entire line, hit the "@" key or the "DEL" key. To cancel a command after it has been entered, hit the "DEL" key.

CAUTION ITEMS:

- ◆ After unpacking your system, allow it to reach normal room temperature before applying power. Allow at least 24 hours after receipt for environmental stabilization to prevent damage due to condensation. Extremes of temperature or humidity can damage the unit.
- ◆ Static discharges can easily damage electronic components. Do not handle the circuit boards unless absolutely necessary, and then only for as short a time as possible. Do not wear static producing clothing. Before you touch a circuit board, discharge your body by touching the system chassis first. And please, don't hand boards around from person to person.
- ◆ Avoid physical shocks, which might cause the Winchester's read/write head to hit the surface of the media. Do not move the system while power is on. When power is off, the heads are brought to a "safe" zone.

Some steps include a list of what to do in case of trouble. If your problems persist after making the indicated checks, call Heurikon Customer Service or our Service Department for help.

1.1 Winchester Booting

<u>Condition/Prompt</u>	<u>Operator Response</u>
Some LEDs come on Hbug prompt, ">"	Turn Power on Push Reset PB Wait 30 sec for drive stabilization
"Standalone boot :"	^bw^
"...RETURN to start at..."	<CR>
UNIX prompt, "#"	<CR>
"...date correct?"	^init 2^
"Enter correct date:"	^n^
"...date correct?"	MMDDHHMM
"...check file systems?"	^y^
"Phase 1, 2, 3, 4, 5..."	^y^
"login:"	(logname)
"Password:"	(password)
"TERM 925"	^dumb^
UNIX prompt, "%" or "#"	

"<CR>" means enter only a carriage return

Table i. Quick Summary - Boot Procedure

[1] Apply power and push the system reset button.

⚠ CAUTION. Be certain your system is not at an extreme temperature. All parts of the system, including the internal drives, must be at normal room temperatures before applying power.

[2] When power is applied, or after a hardware reset, the HK68 is executing instructions from the Hbug monitor pROM. It is generally a good idea to manually push the system reset button after applying power, even if you have the Hbug prompt, just to be certain that all I/O devices are properly initialized. The prompt from the Hbug monitor will look something like this:

```
Heurikon Corporation
Hbug
Ver x.y
>
```

Hbug will allow you to perform a number of simple tasks. Here is a very abbreviated list of Hbug commands:

```

`uc`      Print HK68 Configuration
`um`      Perform RAM test
`dm adrs` Display Memory
`sb adrs` Substitute Byte at adrs
`c adrs`  Call Routine at adrs
`bw`      Boot from Winchester
`bf`      Boot from floppy (M10, SBX-FDIO)
`bsf`     Boot from floppy (SCSI)

```

Refer to the Hbug manual for a complete listing of the commands and more details. If you do not receive a prompt after pushing the reset button, check these items:

- ⊕ The power switch must be all the way in the ON position, the line cord must be firmly inserted at both ends, the fuse must not be blown and the line voltage selector (near the fuse) must be set correctly.
- ⊕ The fan must be working. Do not operate the unit if the fan is not functioning properly.
- ⊕ If the fan is running, push and release the reset button and observe the status LEDs on the HK68. The "S" (Supervisor) LED near the corner of the HK68 or on the front panel should be on. The "S" LED means that Hbug is running. The "H" LED should go on only when the reset button is pressed.
- ⊕ If the "H" (Halt) LED is always on, then the HK68 is not executing instructions, which is indicative of a loose board on the bus or a hardware problem on the HK68. Remove all boards from the system, except for the HK68, and try running the system again. If that does not help (H LED still on), then remove the HK68 and check that all chips are firmly in place. CAUTION: Static discharges can damage components on the HK68.
- ⊕ Check that the serial cables are properly attached to the HK68. Also check that the external cables are connected to your terminal correctly. Try your terminal on the other serial ports. Refer to section {16} and the appropriate hardware manual and drawings for details.
- ⊕ Check the setting of your terminal options. Unless you made a special order, the monitor program will be transmitting the prompt at 9600 baud. If you get some characters, but they are garbage, your baud rate is probably incorrect.
- ⊕ Check that the RS-232-C control signals (RTS, CTS, DTR, DSR) are connected properly. The HK68 expects RTS and DTR to be "true" (positive voltage). Check that these pins on the 25 pin "D" connector at the HK68 are wired correctly:

<u>Pin</u>	<u>Connection</u>
2	Transmit data, from the terminal.
3	Receive data, to the terminal.
4	RTS, from the terminal or connect to pin 5. Must be true. (Not required on HK68/V20)
5	CTS, to the terminal. True if Hbug is running.
6	DSR, to the terminal. True if Hbug is running.
7	Ground
20	DTR, from the terminal or connect to pin 6. Must be true. (Not required on HK68/V20)

See section {16} for more information about the serial ports.

- [3] At this point, allow sufficient time for the Winchester drives to get up to speed. Depending on the type of drives you have, you should hear a series of clicks or whines as the drives reset themselves. Wait at least 20 to 30 seconds for the drives to stabilize.
- [4] After you have the Hbug prompt (">"), enter `^bw^`. This will load a short bootstrap program from the Winchester. Use `^bf^` to boot from a floppy. UNIX will respond:

```
Standalone boot
:
```

Some versions of Hbug have an autoboot feature. If this feature is enabled, then you should not expect the boot prompt; go to step 8.

If you do not get a response, try these checks:

- ⊕ Reset the system and enter `^uc^`. This will cause Hbug to print out a summary of the current hardware configuration. This information may help you locate the problem.
- ⊕ Check that all cables are properly attached to the HK68 and the Winchester controller.
- ⊕ If you do not hear the drive resetting itself after power is applied, and if the system will not boot, then check all of the Winchester drive and controller power and data cables. See the appropriate hardware manual for cable details.
- ⊕ If you get a drive error, wait a few seconds, push reset and try the `^bw^` command again. The drives must have sufficient time to spin up before they may be used. Sometimes, depending on the previous head position, it takes two attempts to boot.

- ✦ An "ID not found" error indicates that the format information on the drive is not readable. The system will have to be booted from floppy. Refer to sections {1.2} (floppy) and {12} (rebuilding) for details.

[5] Enter a carriage return to continue the boot process. This will cause the (default) "/unix" file to be loaded and executed. An input of the form `~/another.unix` can be used to specify a system file other than the default on drive "w0b".

- ✦ If you get a message like "unix not found", the loader cannot find the default /unix file. Reset the system, repeat the above steps and, instead of just hitting the carriage return at this point, try responding `~/unix.tp`, `~/unix.no_tp/`, or the complete path name for any unix file you have on your system. This will instruct the boot program to load a different version of the kernel.

If the boot or kernel programs on the Winchester are corrupted, it may be possible to boot from floppy and correct the problem. If this does not work, then it may be necessary to completely rebuild the system according to section {12}.

- ✦ A "unix not found" error could be due to the root file system not being at the expected block on the drive, possibly because the drive has been configured with a larger swap space. Try responding with `wd(0,16000)`. (This is automatically done in release 7a and later).
- ✦ If you get a "bus error", you probably have a hardware problem, such as bad memory. The bootstrap or file system on the Winchester could also be damaged, in which case you should be able to boot from floppy.
- ✦ Winchester error codes are described in section {14.1}.

[6] The loader will display some messages during the boot process. This information consists of the sizes of the text, data and bss areas of the kernel. This step will take about a minute.

- ✦ Then, you may be asked to "Type RETURN to start at 0x1000". If you are so asked, enter another carriage return to start execution of the kernel.

[7] The kernel will begin by initializing the various I/O devices. You will get a whole bunch of configuration messages, including the amount of free memory (e.g., "mem=765952M"). Allow another half minute for this step.

- ✦ Trouble here usually indicates a memory or peripheral board problem. UNIX is trying to size memory by looking for

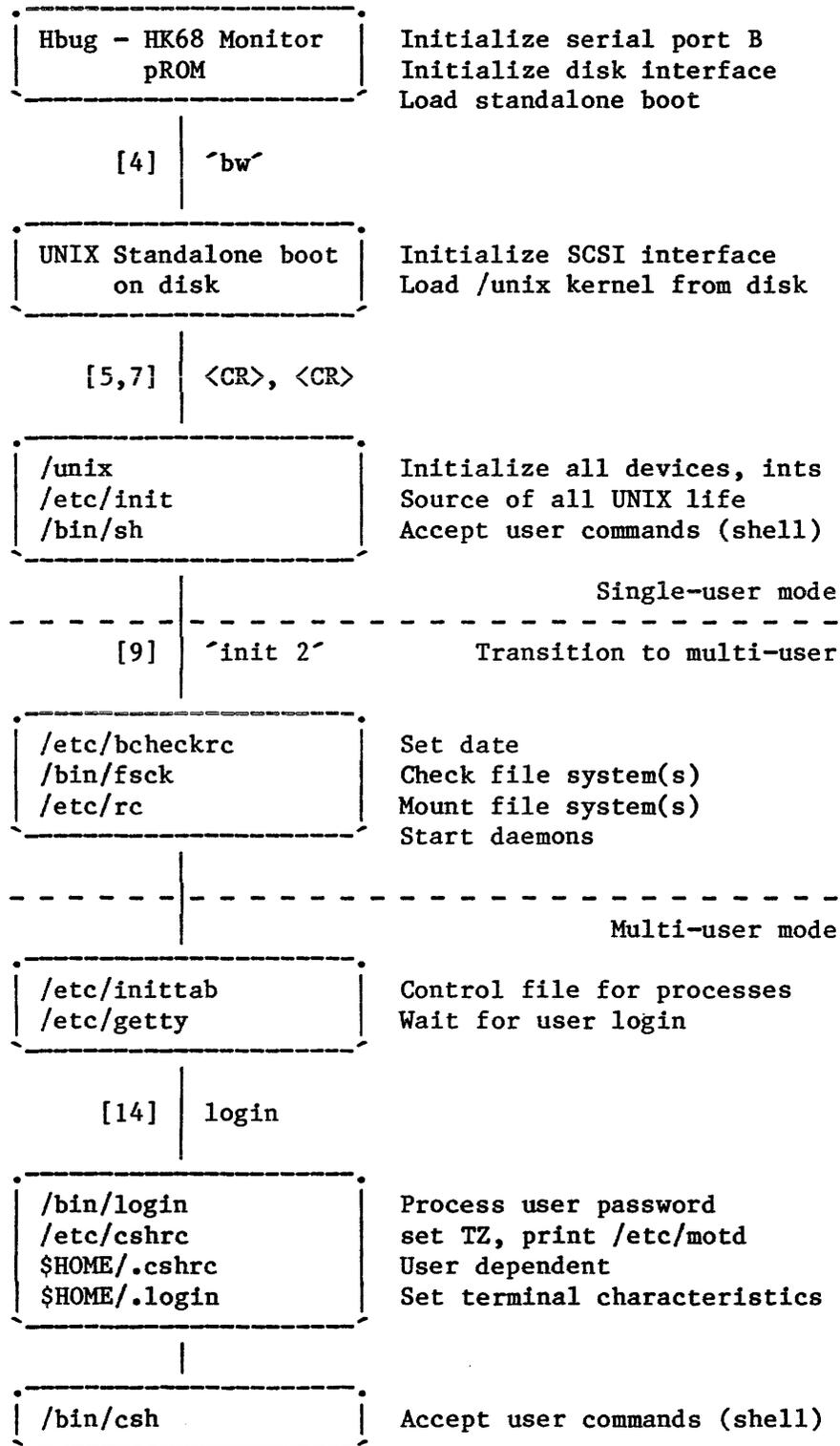


Figure 1. UNIX Startup Sequence - Flowchart

unresponsive or "stuck" bytes (meaning it has hit the end of contiguous RAM), and initialize the I/O devices. If it cannot do this properly, it may or may not recognize the problem and generate a specific error message. Be sure all boards are in their proper slots and that all option switches and jumpers on the peripheral boards are set correctly.

- ⊕ If your terminal is not configured properly, the system may appear to hang. Check the parity controls of your terminal; configure for eight data bits, no parity.
- ⊕ If the system appears to stop after initializing the I/O devices but before the "single user mode" message is displayed, it may be due to the /dev/syscon device node being linked to the wrong tty port. Try hitting the "DELETE" key on the console. If that doesn't help, refer to section {10.10}.
- ⊕ If there is a long delay (about five minutes or so) during the boot phase, it may be because your console terminal does not have DTR and RTS true. Refer to section {16.4}.
- ⊕ If /etc/init is missing or damaged, you will probably not get a prompt, either. Boot from floppy to check or restore the disk copy of /etc/init.

- [8] You should receive the standard UNIX super-user prompt "#". This indicates that the UNIX shell (the command interpreter) is ready for a command.

At this point, you are the "super-user" and are in single user mode. The super-user has access to all files, commands and devices, regardless of the permissions or usual restrictions on them. Because of this, the system is "vulnerable". Therefore, we recommend that the console device be located in a secure area if you have sensitive information stored in your system.

You may want to stay in single-user mode if you are trying to repair damage to the file system or prevent file changes (by another user or by /etc/cron) during a dump. It is not good practice, however, to be super-user all the time, since it is easy to accidentally damage the system by deleting or altering important files.

- ⊕ The visual editor, "vi", will not work in single-user mode unless you manually set the TERM environment variable to your terminal type. See section {3.5} for details. You can use the other editors, however.

- [9] Enter `^init 2` to switch to multi-user mode. In UNIX, the "init" program starts all other programs. Before allowing other users to log on, the init program runs a series of command scripts which checks the date, checks file system integrity, starts background

daemons (e.g., the "cron" program) and removes old log and program locking files. These scripts are described in more detail later in this guide.

- [10] You will be asked if the date and time are correct. They probably aren't, so reply n. You will then be able to enter the correct date and time using the following format:

```

MMDDHHMM
or
MMDDHHMMYY
where MM is the month (01-12)
      DD is the day (01-31)
      HH is the hour (00-23)
      MM is the minute (00-59)
      YY is the (optional) year (85-99)
e.g., 01251330 would be
      1:30 pm on Jan 25, current year

```

It is good practice to set the date and time before entering multi-user mode, since doing so later could burden the system. If the date or time is changed in multi-user mode (while /etc/cron is running), cron will try to "catch up" by doing, at once, all of the tasks it should have done between the old and new times.

- [11] Next, you will be asked if you want to perform a file system check, "fsck". Respond y. Fsck will do a file system consistency check. We recommend that you always answer y. Otherwise, if the file system is corrupted, using it may damage it beyond easy repair. Fsck will offer to repair any file system problems it detects. This procedure can take from five to ten minutes. The standard output from this command will look like this (some numbers may vary):

```

** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Free List
1833 files 22107 blocks 77455 free

```

Refer to the fsck documentation in the UNIX "Administrator Guide" for more details.

- ♣ If fsck finds a problem, it will print details concerning the trouble (e.g., inode number and file ownership) and offer to fix it for you. Generally, it is safe to respond y to all fsck requests. The fsck documentation contains more details.

For releases prior to V.0 7a or on V.2 systems, if fsck makes any changes to the root file system, it will halt when it is done. If you get a "BOOT UNIX (NO SYNC!)" message, push the

system reset button and restart this entire boot procedure, including a second file system check (which should find no errors).

- ♣ Some fsck messages are only warnings. For example, a message of the form "Possible File Size Error..." is probably not an error. Do not be concerned if you see one of these. They may be suppressed by modifying the fsck command used in the startup script.

[12] The system "nodename" will be checked against a value in /etc/bcheckrc. The nodename is used by the uucp logic as a host name; if it is not correct (as would be the case after installing a new /unix file), you will be asked if you would like it fixed.

[13] If /etc/bcheckrc finds no problems, /etc/rc will be run which will start the standard UNIX daemons (such as update and cron) and mount any secondary file systems (such as /dev/wlb). These actions may take up to 30 seconds. Then, you will get the standard UNIX "login:" prompt, usually preceded by a banner of some sort, which identifies your computer system.

[14] To login, enter a valid user name, such as `rootcsh` or `guest`. The `who` command can also be executed at this point, without logging in, simply by entering `who`, assuming the /etc/passwd file has not been altered to prevent this. The console device usually logs on as `rootcsh`.

- ♣ The "rootcsh" super-user login will use the C-shell, "csh". The Bourne shell login, "sh", is `root`. Use `root` if you have trouble logging in as "rootcsh". Refer to section {7} for information on the differences between "csh" and "sh". A super-user login should be used only for system administration functions, not for normal usage. Also, it is best to use a regular user login, then history files used by `last` and `su`.

- ♣ Other login names may be used if they have been setup previously in the /etc/passwd file by your system administrator, per section {10.1}.

[15] If there is password protection for the login name which you use, the system will request your "password:". Enter your correct `password`. Be sure to hit the carriage return key, too. The system will not echo the characters you type.

[16] You will see the "message of the day", which is contained in /etc/motd.

- ♣ "Login incorrect" means that you have either chosen a login name that does not exist in /etc/passwd or you have not typed the password correctly. Be sure you are using the proper case,

since UNIX makes a distinction between upper and lower case characters, even in a password. Also, while entering the password, you cannot correct typing errors with the backspace key. You must use the "@" and retype the whole password. Or, enter a carriage return and wait for another login prompt.

[17] The operation of the system from this point on is highly dependent upon the contents of the /etc/cshrc, ~/.cshrc and ~/.login files, which are associated with the particular login name you used. ("~" means your home directory.)

[18] You may be asked what type of terminal you are using via a message of the form "TERM (925) ". If you are using the type indicated (e.g., a Televideo 925), simply enter a carriage return. If you are using another type, enter the terminal type code. For example, `^avp^` would be an ADDS Viewpoint. This will cause the proper initialization sequence for your terminal to be used, instead of the default. It is important for the terminal type to be set properly since the visual editor ("vi") will not work correctly otherwise. (The line editors "ex" or "ed", however, will still work.)

♣ If you do not know the proper abbreviation for your terminal, enter `^dumb^` and you will be minimally initialized. Later, you can scan through /etc/termcap for the correct entry, or use "ed" or "ex" to add a new terminal type to the termcap file. You can also modify your .login file to make the terminal selection automatically for you, once you know the correct abbreviation. The termcap file contains entries for over 200 terminal types.

[19] At this point, you should be "on-the-air". If you logged in as "rootsh" or "root", you should have the super-user prompt, "#". If you logged in as a normal user, then you will probably have a "%" prompt. These are the default prompts; they can easily be changed. You can have your prompt automatically set when you log in by including the appropriate command in your .login file. See sections {6.5.2} and {7.2}.

[20] Look in section one of the UNIX User's Manual for descriptions of the UNIX commands. See section {4.1} of this guide for help in using the UNIX manuals, section {8} for some simple command examples and section {18} for a list of suggested reference books.

♣ If all else fails, read the instructions!

[21] Some commands or functions are restricted, unless you are the "super-user". To become the super-user without logging off, enter `^su rootsh^` and, when prompted, the rootsh password. Later, to return to your previous login, type `^exit^`.

♣ Do not run commands as the super-user any longer than necessary, so as to lessen the chances of accidentally deleting or damaging important files.

[22] To terminate your UNIX session, type `exit` or `logout`.

System power down procedures are detailed in section {2} of this guide.

Certain features have been "preset" at the factory. The initial system configuration information can be found in section {1.3}.

1.2 Floppy Booting

These procedures may be used instead of the standard method of booting from the Winchester. Typically, the only reason you would boot from a floppy would be if you are rebuilding your system or trying to recover from a "crash" which wiped out the standalone boot program or a critical file on the Winchester.

These procedures assume some familiarity with the system. Refer to the previous section on booting from the Winchester for help concerning the expected UNIX responses and troubleshooting hints if you encounter problems.

- [1] Push the system reset button.
- [2] Insert the floppy disk labeled "boot" and close the door. This disk should be write protected.
- [3] Enter `bf`.
- [4] When you get the standalone boot prompt (":"), remove the boot diskette and insert the "rebuild" diskette.

♣ If you did not get the standalone boot prompt, remove the diskette and check that it has been properly inserted.

- [5] Hit the carriage return key, which will cause the "/unix" file to be read from the floppy. This will take a few minutes to complete.
- [6] When you get the UNIX prompt, "#", you may execute any command which is contained on the diskette. Not all commands will be available, due to limited space on the diskette.
- [7] If you are going to rebuild the system, refer to section {12}. Note: Rebuild the system only as a last resort.
- [8] If you want to examine the Winchester, enter `mount /dev/w0b /floppy`. The "/floppy" directory on the floppy based system is really just a dummy directory (it is empty). Mounting the Winchester there is a way of attaching the Winchester to the floppy

root file system so that it can be accessed. You can then refer to the Winchester files with the pathname `~/floppy/name...`. See section {12.9} for additional information.

If you cannot boot from the Winchester and you feel that the Winchester's file system is okay, then you may have a defective boot program (which is on the first few sectors of the Winchester) or the `/unix` file on the Winchester could be bad. Either of these conditions can be corrected without rebuilding. Section {12.4} explains how to rewrite the bootstrap. See {11.1.3} for help in recovering `/unix`.

1.3 Initial System Configuration

Certain system features have been configured at the factory, as shown below.

Feature	Initial State	Files/Commands	Ref Section
uucp	nodename = "hum"	<code>/etc/bcheckrc</code>	16.8
	uucp daemons disabled (3)	<code>/usr/lib/crontab</code>	6.8
cu	<code>/dev/tty2</code> at 1200 baud	<code>/usr/lib/uucp/L-devices</code>	16.9
lp	enabled	<code>lpadmin(i)</code>	16.10
	lpsched running	<code>/etc/rc</code>	
	printer "tosh" on <code>/dev/tty0</code>	<code>lpadmin(l)</code>	
lpr	off. <code>/dev/lp</code> not created	<code>lpr(l)</code>	16.10
pass- words	no passwords required	<code>/etc/passwd</code>	6.1
logins	console speed = 9600 baud	<code>/etc/gettydefs</code>	6.3
	alternate port = <code>/dev/tty1</code>	<code>/etc/inittab</code>	6.2
	modem port = "off"	<code>/etc/inittab</code>	
	modem speed = 1200 baud	<code>/etc/inittab, /etc/gettydefs</code>	
	other serial ports "off"	<code>/etc/inittab</code>	
daemons	cron enabled	<code>/etc/rc</code>	6.4
	calendar, 5:00 am	<code>/usr/lib/crontab</code>	6.8
	atrun twice per hour	<code>/usr/lib/crontab</code>	6.8
accounting	off	(numerous)	4.7

Table 2. Initial System Configuration

2. POWER-DOWN PROCEDURE

- ♣ We do not recommend that you power the system down if you will be using it again in only a few hours. It is much less traumatic for the equipment to be left on over a weekend than it is to turn power off and on. Also, the UNIX time of day clock and /etc/cron features should be allowed to run normally, so the automatic (preprogrammed) system maintenance functions can operate as scheduled.
- ♣ If you plan to move the system, it is advisable to power it down. The media can be damaged if physical shocks cause the Winchester read/write head to touch the disk surface.

Before you reset the system and remove power, it is important that UNIX has written all of its buffers to the Winchester drive. UNIX does not necessarily do a physical disk write at the same time that a program executes a logical "write" system call. If care is not taken, you may lose portions of data which you thought were written to the disk, but weren't. Also, UNIX maintains information in memory concerning the file system (the "super-block") which must be written to the drive before the system is stopped.

- [1] Log off all users but one, usually the console. This step is not strictly necessary, but it insures that other users know the system is coming down. In addition to being courteous, this prevents users from making file changes just prior to stopping the system.

You can do this by entering `^exit^` at each terminal or by using the "ps" and "kill" commands at the console. You could also use "wall" to notify all users to logoff and wait for them to do so. Another alternative is to enter `^init s^` to force a return to single user mode. See section {10.10} for more information about the "init" command.

- [2] Enter `^sync^`.
- [3] Enter `^sync^` (again). This insures that all UNIX internal buffers are flushed out to the drives. Some people think the second `^sync^` command is just superstition. Others explain that the second command insures the first has completed. Anyway, two or more do no harm.
- [4] Remove all floppy diskettes or cassette tapes from the drives.
- [5] Push the system reset button, then power the system down.

3. IN CASE OF TROUBLE

3.1 Cannot Boot

If you cannot boot the system from the Winchester, try to boot from a floppy, as detailed in section {1.2}, above. If the standalone boot program or the /unix file is damaged on the Winchester, you will be able to repair the problem without too much trouble. If there is a more severe problem with the file system on the Winchester, then it may be necessary to rebuild it; but, do so only as a last resort since the rebuild procedure is time consuming and will wipe out any recent changes made to your files.

Read the troubleshooting hints given with the instructions on booting the system at the beginning of this guide.

3.2 Programs that Don't Work

Sometimes a previously functional or an original distribution program will refuse to work, no matter how hard you try. Sometimes the error message will be of help, sometimes not. Here are some debugging hints:

- ♣ Do you have permission to execute the program? Are you the owner of the file, a member of the group or just "other"? Is the appropriate execute bit on? Use `ls -l pgmname` to check these bits. Change to super-user and try it again. Refer to `ls(1)` and `chmod(1)` in the UNIX User's Manual.
- ♣ Do you have permission to search the directories in the program's path? Check the "x" bits using `ls -l`. Change to super-user and try it again.
- ♣ Do you have the permissions on your devices set correctly? For example, the "ps" command is "suid check". In order for it to work, the /dev/mem and /dev/swap devices must be owned by "check". The floppy or tape devices must be mode 666, or you must be running as super-user, to do floppy disk or tape I/O.
- ♣ If the "ps" command produces no output, garbage output or complains about a bad "namelist", then you probably booted a kernel other than /unix. Use the "-n" option with "ps" (to specify the kernel file in use) or rename your kernel file to "/unix", if appropriate.
- ♣ Do you have to be the super-user to execute the program? Change to super-user (via `su rootcsh`) and try it again.
- ♣ Some files are "hidden" from the "ls" command. If the file name begins with a dot, e.g., ".cshrc", then you have to use `ls -a` to list them. Dot files are not included in the list when the shell expands a wild card, such as "*".

- ♣ Are you using the correct command format? Check the UNIX User Manual. Try a simpler version of the command.
- ♣ Do all the files and directories that the command needs exist and do they have the proper r/w permissions? For example, "/bin/mail" needs the /usr/mail directory with 777 permissions. Check the "FILES" section of the manual page and verify that all files and directories are present.
- ♣ Have you just "installed" the program somewhere, like in /usr/bin? Execute `rehash` to make it visible to your csh.
- ♣ Is there another program of the same name earlier in the shell's search path? Use `ls` to search for such a program or the "whereis" script, shown in figure {20}. Enter the full pathname of the program, as in `/usr/yourname/a.out` or, if the program is in your current directory, `./a.out`, instead of just `a.out`.
- ♣ Maybe there is a csh "alias" for the bad command. Type `alias` to see the current set of aliases. You can use `unalias cmd` to remove an alias.
- ♣ Is the program in your search path? Enter `echo $path` (csh) or `echo $PATH` (sh) to see the path list. You can change the path list by putting a "set" command in your .login file.
- ♣ If the program is binary, try `strings /bin/pgmname` and check for undocumented files, devices, directories or other programs it is trying to access. This step is not easy to explain since you must use some intuition to figure out what the program is trying to do. To get the hang of this technique, experiment by running "strings" on some familiar programs.
- ♣ If the program generates a "core dump" and all other things have been checked, try reading another copy of it from the appropriate distribution diskette or tape. You can check if the version you have is okay by using the `sumdir` command, and comparing the results with the appropriate line in /etc/vchk_tree for that program.
- ♣ Does your command line have any special shell characters? You may have to escape the special characters using a pair of single quotes (') or a backslash ("\"). See section {7.5}.
- ♣ If you are trying to run a shell script, it may not be executing with the proper shell. If the script is started from a Bourne shell, then it will execute using a Bourne shell. However, if you are invoking the script from a C-shell (as is most likely), then the first character of the script is used to select the shell for running the script. If that character is "#", then a C-shell ("csh") will be used; otherwise, a Bourne shell ("sh") will be used.

3.3 Terminal or System Lockups

If your terminal stops functioning, don't panic, and don't push the reset button. The reset button is always the last resort. Try these steps to clear the problem:

- [1] Enter `^Q` (Control-Q) in case you have stopped your output with a `^S`.
- [2] If you are in the "vi" editor, hit ESC twice and try the `^R` command. If the screen does not update, enter `^ZZ` to save your file and exit the editor. Do this in case the system is still acting on your commands, even though you do not see a response.
- [3] Hit your `^DEL` key to kill the program. If that doesn't work, try entering `^_|` (Control-|) to send a QUIT signal to the process.
- [4] Enter `^<LF>reset<LF>` (that's "reset" surrounded by two line feeds). The line feeds are used here in case the tty modes have been set to prevent a carriage return from terminating a line.
- [5] Enter `^stty sane`.
- [6] Enter `^tset`. Wait 10 seconds.
- [7] Hit carriage return a few times. See if you get a prompt.
- [8] Enter `^exit`. Wait 10 seconds. Maybe your shell is still listening.
- [9] Reset your terminal (power it off, then on). Repeat the above steps.
- [10] Go to another terminal, login as "rootcsh", execute `^ps -e`, find the programs attached to the locked port and kill them. Use `^kill -9 pid`. If you only have one terminal, reconnect it to another serial port. This is why we recommend having gettys on unused ports (see section {6.2}).
- [11] If all of the above fails, now is the time to reset the system. But before you do, enter two "sync" commands at a working terminal, if there is one, or wait 60 seconds. Then, reset the system. (If the system is still running, the /etc/update process will do an automatic "sync" every 30 seconds.)
- [12] Reboot and be sure to run `^fsck`.
- [13] If you were in the editor when the system crashed, you can recover the edit session up to the last two or three changes you made to the file. Refer to sections {9.3}, {9.4} and the UNIX manuals for more information about the UNIX text editors.

3.4 System Crashes

If the system "crashes", it is usually due to a hardware error. When you are executing a user program (without super-user privileges), it should be impossible to accidentally do anything which would bring the system down. If you tried to do something illegal, the worst you could get would be "Segmentation violation -- core dumped." Such a message usually means the program made a memory reference to an illegal (out of bounds) address.

A true crash results in a "panic" message from the kernel or the "H" (halt) LED coming on. A panic message means that the kernel encountered a situation which is impossible to resolve. So, it just stops. This could be the result of a memory error, where some value the kernel read was way out of limits. At any rate, panic errors are indicative of a hardware problem or an obscure software bug. Contact us if you have persistent errors of this nature.

Refer to section {3.6} for information on how to report bugs.

There are two commands in the Hbug monitor program which may help you localize a hardware problem. Refer to the Hbug manual for details on the "uc" and "um" commands.

3.5 Other Problems

- ♣ (M10, V10 only). If the console scrolls up one line every few seconds, or if you get strange status messages from one of the device drivers, check to be sure all of the user jumpers on the HK68 are removed. (See section {15.3} for more information on the user jumpers.)
- ♣ If you try to send data to a serial port to which there is no device connected, the program might hang, waiting for the characters to be transmitted. The HK68 will not transmit unless DTR is true. Even a `kill -9 pid` may not release the process. Connect a serial device (your terminal will do) to the port for a few seconds. This will allow the HK68 to transmit the characters, so the process can terminate.
- ♣ If you delete a file using the "rm" command, it is gone. There is no practical way to recover a deleted file since the disk blocks are immediately released for reuse. You might be able to reload a lost file from a backup tape or diskette. You may be interested in using the "rm" alias shown in section {7.1}.
- ♣ The serial port characteristics may not be set properly for your terminal or personal tastes. The `stty` command will let you change some of the special characters, such as backspace, kill and end-of-file. You can use `stty` to change the baud rate, specify parity options and select how the backspace key affects your CRT display. Refer to section {16.11} for more information.

- ♣ The "console" port, which, by default, is connected to hardware port B, may be redefined to point to a different hardware port, by changing /dev/syscon. This could be done automatically by "init", if you switch to single user mode, in which case you may have unexpected trouble when rebooting. See section {10.10} for details.
- ♣ If the file system becomes full, due to unmanaged log files or too many old files, you can boot from floppy, mount the winchester and delete some files to create space. See section {12.9} for details. There is some discussion in section {10.4} about monitoring file system usage and obtaining more disk space.
- ♣ The visual editor, "vi", will not work properly unless your terminal type is set in the "TERM" environment variable. When you login in multi-user mode, commands in your .login file set the TERM and TERMCAP variables. If the TERM variable is incorrect or if you are in single-user mode (where TERM is not set), then you may set the TERM variable manually, as follows (this example is for an ADDS ViewPoint terminal, which is coded "avp" in /etc/termcap):

Single-user mode (sh): TERM=avp; export TERM
Multi-user mode (csh): setenv TERM avp
- ♣ If you are having trouble sending mail between machines, check sections {10.8} and {16.8} for help.
- ♣ UNIX is a very flexible operating system. Sometimes, apparent malfunctions are actually not problems at all, but simply normal operation of the system as it is presently configured. Chances are, you can modify the behavior of your system to suit your needs; it only takes experience to learn how. Also, pranksters occasionally prey on UNIX neophytes. A common trick is to change the shell prompt to look like an error message.
- ♣ To avoid hangups when using a defective floppy diskette, specify the "raw" device instead of the "block" device.
- ♣ If you run out of swap space (usually indicated by "Not enough swap space to fork" messages), you may need to enlarge the swap space. Refer to section {12.7}. A short term solution is to add memory or reduce the number of users.
- ♣ If you suddenly run out of memory to run programs (e.g., the compiler starts to complain) you may really have too little memory or a previous program which used shared memory may not have released the shared memory area. Refer to section {15.14}.
- ♣ If the date looks right, but the day of the week is wrong, check the year.

3.6 Reporting Bugs

If you do have a repeatable system software error, please contact our Customer Support Department with the following information:

- ♣ Your UNIX (or other program) serial number, as assigned by us when shipped.
- ♣ The output of `size /unix` and `sumdir /unix`.
- ♣ A list of any local changes you have made to the system, such as the addition of a custom device driver. We may ask you to demonstrate that the bug exists on an unmodified version of our UNIX release.
- ♣ The program size, checksum and version (output of `size`, `sumdir` and `version` programs).
- ♣ The shortest version of your program which demonstrates the problem. If we can repeat the error, we can correct the bug.
- ♣ The output of any register dumps from the console. In the case of a kernel panic, copy down the register data before rebooting.

4. THE UNIX MANUALS

One key to understanding UNIX, and learning how to use all the features, is to thoroughly understand how the UNIX manuals are organized. This will make it easy for you to find information and to dig out buried items.

- ♣ A manual is a fundamental reference tool. It contains precise information about commands and UNIX features.
- ♣ A guide is a tutorial or description of a feature or program. A guide gives more details about certain programs and has usage examples.

This document is a guide, and there are some manual pages at the end.

4.1 User's Manual

The User's Manual is the primary reference manual for UNIX. There are six "sections" in this manual, split between two books. The first book contains the user portion of section one, the second book contains sections two through six. Section one is the primary user reference. It contains the command descriptions.

Usually, a separate page is reserved for each command. If a group of commands are related, they may share pages in the manual. The pages are ordered alphabetically (by the first command in a group) with the command clearly listed at the top of each page.

At the beginning of each book, there is a "permuted index." These are, perhaps, the most important pages of the manuals. At first, it may look a bit complicated. But this is due to its unusual format; a format which actually provides much more information than a typical "index". Here are some sample lines from near the middle of the permuted index:

```
lpstat: print  LP status information . . . . lpstat.1
              lpr: line printer spooler . . lpr.1
information.  lpstat: print LP status . . . lpstat.1
directories.  ls: list contents of. . . . . ls.1
              update. lsearch: linear search and. . search.3c
              pointer. lseek: move read/write file . seek.2
              m4: macro processor . . . . . m4.1
              m4: macro processor . . . . . m4.1
              tp: magnetic tape format. . . . . tp.4
mail, rmail: send mail to users or read mail. . mail.1
program. ctags: maintain tags file for a C. . ctags.1
```

Figure 2. Permuted Index

To use it, scan down the middle column, which is in alphabetical order. Every command, system call, subroutine and file page in the manual starts with a "NAME" section containing the command or routine name followed by a

UNIX User's Manual

Permuted Index
Section 1
Commands & Programs

UNIX User Guide

Basics
C Shell
Bourne Shell
Editor Tutorials

UNIX User's Manual

Permuted Index
Section 2
System Calls
Section 3
Subroutines
Section 4
File Formats
Section 5
Misc Facilities
Section 6
Games

UNIX Programming Guide

"C" Tutorial
UNIX I/O System
UNIX Implementation

UNIX Support Tools Guide

Make Tutorial
SCCS, dc, lex, yacc
Awk Tutorial
Uucp description

Document Processing Guide

Nroff/troff Tutorials
Macro Descriptions

UNIX Administrator's Manual

Permuted Index
Section 1
Commands & Programs
Section 7
Special Files
Section 8
System Maintenance Procedures

UNIX Administrator Guide

Fsck Description
Line Printer Spooler
Uucp Administration

Heurikon UNIX - Ref Guide

Contents

You are here --> *

Special Manual Pages

Index

Figure 3. UNIX Documentation

very brief description. The permuted index is formed by taking each of these lines (one from each manual page) and rotating it numerous times so each keyword in the description has a turn at being positioned at the middle column. The result is alphabetized. Thus, each command may be listed in the index in more than one place, depending on the keywords used in its description. This scheme makes it easy to find almost any command, as long as you know something about its function.

The right hand column of the index lists the command name under which the information can be found. The suffix number following the name indicates the section number. Thus, the "ls" command, which is listed in section one of the manual as:

```
ls: list contents of directories
```

will be cross-referenced in the permuted index at "contents", "directories", "list", and "ls".

Frequently, the suffix number is enclosed in "()", as in "mail(1)" and "seek(2)".

Commands and files suffixed ".lm", ".7" or ".8" are in the "Administrator's Manual", described later.

4.1.1 Section 1 - Commands and Application Programs

These are the descriptions of the commands which can be invoked by a user in response to a UNIX shell prompt. This is the most used section of the manuals. Each page contains a detailed list of the options and arguments which may be used with the command. Most descriptions also include an example of how to use the command. The "SEE ALSO" section will point you to other related commands or files, the descriptions of which frequently shed more light on the usage of the command.

Most of the actual programs are located in the /bin and /usr/bin directories.

If you cannot find a command in this section, which you know exists, look also in section one of the the "Administrator's Manual", described below.

4.1.2 Section 2 - System Calls

System calls are the means which a program uses to communicate with the UNIX kernel to request I/O, allocate resources or do other kernel functions. This section describes the system calls and details the exact format for both "C" and assembler interface with the kernel. Example system calls would be disk file "open", "read", "write" and "close".

The introduction found at the beginning of section two explains all the possible errors which could be returned from a system call and defines some of the basic terminology used by UNIX. You should skim the introduction (intro.2) even if you do not plan to do any "C" programming under UNIX.

4.1.3 Section 3 - Subroutines

There are many pre-coded subroutines which may be called from a "C" program. For example, to print a formatted line containing some text and a decimal value, you need only use a statement of the following form:

```
printf("The value is: %d.", value);
```

The "printf" subroutine will take care of all the dirty work associated with printing the text and converting the value. The above command would output a line like this:

```
The value is: 349.
```

Don't confuse a subroutine with a system call. A subroutine performs some useful or commonly used function which may or may not make a system call. A system call makes some specific and primitive demand on the UNIX kernel.

4.1.4 Section 4 - File Formats

There are numerous control files used in UNIX. Most of them are in ASCII, so they can be read by us humans and easily modified. Section four details the formats of particular file types, such as:

- ♣ The System V.2 terminfo data base.
- ♣ executable files (a.out) and core image dumps (core)
- ♣ archive libraries (ar) and tape archives (cpio)
- ♣ the file system (fs), directories (dir) and inodes (inode)
- ♣ process control files (inittab, gettydefs)
- ♣ the system user information and password file (passwd)

4.1.5 Section 5 - Miscellaneous Facilities

This is a small catch-all section. The most important item contained here is the definition of the terminal capabilities data base file (termcap).

4.1.6 Section 6 - Games

This section explains how to run the games and educational programs. They could have been included in section one; but, since some installations might want to delete them or restrict their use, they are documented separately. The actual game programs are kept in a separate section of the file system (/usr/games). The programs listed in this section are not supported. They are provided for recreational use only.

4.2 Administrator's Manual

This is a supplement to the User's Manual described above. The commands and files listed in this volume are generally used only by the system administrator, since their improper use could compromise the integrity of the system.

4.2.1 Section 1 - System Maintenance Commands and Programs

The system administrator can use these programs to configure the system or to check and repair it. Most of the commands in this section reside in the /etc directory. In the permuted index they are suffixed ".lm".

4.2.2 Section 7 - Special Files

UNIX "special" files refer to I/O devices. This section contains information on the "mem", "null" and "tty" devices. There are also descriptions of the networking mail aliases and hosts files in this section, which really ought to be in section four.

4.2.3 Section 8 - System Maintenance Procedures

Additional information on the boot procedure and kernel error messages are listed here, along with the manual pages for some of the mail programs.

4.3 User Guide

This guide is a general overview and a description of some of the features of UNIX. It has a section on "Basics", which, like this document, offers some help to the beginner. It also has tutorial sections on the editors, including "vi", and both standard shells, "sh" and "csh". You should definitely read the sections on "vi" and "csh" before going too far with UNIX.

4.4 Programming Guide

If you plan to write any "C" programs, then this is the place to go for some tutorial information on the "C" language and "C" libraries. This guide also contains the (only) detailed description of the "UNIX I/O system" (which is necessary information if you plan to write a device driver) and "UNIX Implementation" (with details on the file system, I/O system, and some aspects of the kernel).

If you are going to do any serious "C" programming, you must get the the ultimate "C" reference book:

- ♣ The C Programming Language, by Brian W. Kernighan and Dennis M. Ritchie. Prentice-Hall.

4.5 Support Tools Guide

There are many software "tools" in the UNIX system. This guide describes a number of them, using tutorials. For example, you will find details on "dc", "lex", "awk", "make", "yacc" and the "SCCS" package. There is also a section on the "uucp" facility.

4.6 Document Processing Guide

If you anticipate using the document preparation programs, "nroff" and "troff", the Document Processing Guide will answer most of the questions concerning their use. For example, this document was created using the "mm" macros and "nroff". "Nroff" handles all section numbering, page formatting, column justification and other functions associated with the document.

4.7 Administrator Guide

This guide contains information and advice for the person who is primarily responsible for the operation of your system. Most notable are these sections:

- ♣ User mode - has some discussion about fsck.
- ♣ File System Checking - fsck. The system administrator should definitely read this description of fsck. Fsck is executed semi-automatically (by /etc/bcheckrc) before multi-user mode is entered, after a system boot.
- ♣ Line Printer Spooling System. This section explains the installation and operation of the line printer spooling programs. This package allows UNIX to support a number of various printers simultaneously on one system. See section {16.10} for hints on installing and using LP.
- ♣ Uucp Administration. The uucp system allows two computer systems to communicate with each other and exchange files over a serial link, either hard wired or via modems and telephone lines. This section explains the formats used by the various control files and how to use some of the commands which oversee the operation of the uucp system. See section {16.8} for hints on using uucp.
- ♣ System Accounting - description. (Note: Most of our customers will not want to use the system accounting package. The processor overhead and maintenance of the programs themselves are usually not worth the information gained in a small user environment.)

5. THE UNIX FILE SYSTEM5.1 Structure, File and Directory Names

The UNIX file system structure is generally described as a "tree". It has a base, or "root", plus limbs, branches and files. The limbs and branches are the directories. To completely describe the location of a file you need its name (e.g., "test.file") and its path. The path describes the sequence of branches (the directories) which must be traversed in order to get to the file. The root directory is called "/". All subsequent directories are then listed, separated by a slash (another "/") and, finally, the filename is placed at the end. The slashes within such a "pathname" have no relation to the initial "/" used to identify the root.

For example, if "test.file" is located in your home directory (which we assume is named "yourhome"), its complete pathname would be:

```
/usr/yourhome/test.file
```

In this example, the file can be found by starting at the root, moving from there to the "usr" directory, then to the "yourname" directory which will contain "test.file". There are also shorthand notations for specifying a filename. If you are currently "in" your home directory ("yourhome"), then it is sufficient to simply say

```
test.file
```

to specify the file. Here is a summary of the forms which may be used to specify any particular file (assumes csh):

Form	Relative to	Examples
/name	root	/usr/bin/sumdir /usr/guest/test.file
name	working directory	test.file
./name	working directory	./a.out
../name	parent directory	cd ..
~/name	your home directory	cat ~/mbox
~fred/x	Fred's home directory	cat ~fred/mbox

Table 3. file name forms

The "working directory" is the directory which you are "in". You can move around in the directory tree by using the "cd" (change directory) command.

Whenever a filename is required as a command argument, you may chose any of the above methods to specify the name. That is, the shell (and the kernel) will accept either a full pathname, or one which starts from the current working directory. The "~" style is recognized by the C-Shell, so that style cannot be used when issuing system calls to the kernel.

The special names "." and ".." (called "dot" and "dot dot") refer to the current directory and the parent directory, respectively. They allow references to be made relative to the current directory. For example, if you are in /usr/yourhome/src and want to move to /usr/yourhome/bin, any one of these lines will do the job:

```
cd /usr/yourhome/bin
cd .. ; cd bin
cd ../bin
cd ~/bin
cd ; cd bin
```

As used in the last line above, the "cd" command without any arguments simply returns you to your home directory.

Any character can be used in a file name. However, because some characters have special significance to the shell or as option flags, we recommend you use only upper or lower case letters, numbers, underscore and the period. A dash ("-") may be used but, to avoid confusion with command options, not as the first character of a file name.

In UNIX, there are few restrictions when naming files. However, there are certain conventions. The following file types, for example, are frequently encountered (the "*" is a wild card and stands for any sequence of characters):

```
*.c      C source code file
*.s      Assembly language source code
*.o      Object file (binary)
*.h      Include file (header file)
*.f      FORTRAN source code file
*.p      Pascal source code file
s.*      SCCS file
.*       Hidden file (hidden from "ls")
a.out    Default link editor (ld) output
```

5.2 Creating Directories and Files

The following commands may be used to create or modify the directory structure. Files may also be created by redirecting the output of a command, as described in section {7.6}.

```
mkdir    make directory
rmdir    remove directory
rm -r    remove, recursive
mv       move
ln       link
cp       copy
```

5.3 Typical Organization

There is a typical file system layout for a UNIX system, although there are many variations. Some programs assume certain files are in certain directories; so, it is usually a good idea to stay close to the convention.

The following list illustrates a portion of the UNIX file system organization.

<u>Directory</u>	<u>Notes</u>
/bin	Most commands
/floppy	A place to mount a floppy
/lib	Libraries
/etc	System maintenance programs/files
/tmp	Temporary files
/lost+found	Used by fsck to stash lost files
/usr/bin	More commands
/usr/tmp	More temporary files
/usr/adm	System accounting programs/files
/usr/mail	Mail spool area
/usr/games	Game programs and files
/usr/guest	Guest login home directory
/usr/src	System source code
/usr/include	System include files
/usr/spool/uucp	Spool area for uucp system
/usr/spool/lp	Spool area for line printers
/usr/spool/at	Spool area for the "at" command
/usr/lib/uucp	Uucp programs and control files
/usr/lib	More libraries
/usr/lib/cron/	Cron directory (System V.2)
/usr/yourname/	Your home directory
/usr/lib/terminfo/	Curses directory (System V.2)
/usr/yourname/src	Place for your source code
/usr/yourname/bin	Place for your commands
/usr/local/yourname/	Alternate home directory
/usr/local/bin/	Alternate binary directory

Figure 4. Typical UNIX Directory Structure

5.4 Owners and Permissions

Every file (and directory) in UNIX is "owned" by someone, and has a group ownership, too. The owner of a file can place certain restrictions on the use of the file or directory by others. The "ls -l" command will display the ownership information for a file. An executable file can also be given a "suid" (Set User ID) attribute, which means that it can be given special permissions while it is running. These commands are used to check and modify file permissions:

```
ls -l
chown
chgrp
chmod
```

Refer to the UNIX User's Manual for details.

5.5 Repairing a Damaged File System

Sometimes, it is possible to repair a damaged file system. The techniques are not easily described, since the procedures require a thorough knowledge of how UNIX operates. It is, therefore, an advanced topic. Here are some of the commands which are used:

```
ls -i
find
ncheck
fsck
dd
badblk
pstat
fsdb
od
```

6. TOUR OF IMPORTANT FILES6.1 /etc/passwd

The password file is one of the most important files in the system. Critical information about all authorized system users is stored there. This file contains the user's login name, encrypted password, ID number, and home directory. The password file contains "public" data. It has general read permissions, since many UNIX programs use the information to correlate user id numbers with names. There is no security risk, however, because the passwords are encrypted.

```

root:Wz2YUsEt5lSnw:0:0:/:/bin/sh
rootcsh:4zuAHY2G9sHtI:0:0:/:/bin/csh
daemon:xxxxxxxxxxxxxx:1:1:/:
bin:xxxxxxxxxxxxxx:2:2:/:bin:
sys:xxxxxxxxxxxxxx:3:3:/:bin:
adm:xxxxxxxxxxxxxx:4:4:/:usr/adm:
uucp:T7Hl7CubieODM:5:5:admin:/usr/lib/uucp:/bin/csh
check:xxxxxxxxxxxxxx:6:6:/:
who::22:0:who command:/bin:/bin/who
guest:BNBenIFsomjhE:100:100:/:usr/guest:/bin/csh
pete:neCtlikEGi6Dk:101:1:Peter Lee:/usr/pete:/bin/csh
jeff:eS9qfRtUyg1Og:102:1:Jeff Kline:/usr/jeff:/bin/csh
dan:CHxxiv3lH09DU:103:1:Dan Lake:/usr/dan:/bin/csh

```

Figure 5. /etc/passwd file (typical)

All files in the system "belong" to someone. This file relates the file owner ID codes to their names. The system administrator can edit this file to add or delete users. The gobbledygook fields are encrypted passwords. If a user forgets his password, the administrator can delete the password from /etc/passwd. The `passwd` command may be used to enter a new or change an existing password.

The lines with "xxx..." in the password field are for pseudo-users. These "users" are owners of certain system files and directories. For example, most programs are owned by "bin" and the accounting files are owned by "adm". There is no real user named "bin"; the password file entry for "bin" associates user id number "2" with the name "bin".

Note that "root" and "rootcsh" both have the same user id number ("0"). By convention, the super-user is always user id "0". Logging in as "root" or "rootcsh" will give you super-user privileges.

For more details, refer to section four of the UNIX User's Manual.

(Glad to see you're using the index.)

6.2 /etc/inittab

This is a control file for the "init" program. It controls the sequencing used to switch from single to multi-user modes. It specifies which serial ports are to be used for user terminals and the port's baud rate.

```
is:s:initdefault:
bl::bootwait:/etc/bcheckrc </dev/console \
    >/dev/console 2>&l #bootlog
bc::bootwait:/etc/brc 1>/dev/console \
    2>&l #bootrun command
sl::wait:(rm -f /dev/syscon; \
    ln /dev/systty /dev/syscon;) \
    1>/dev/console 2>&l
rc::wait:/etc/rc 1>/dev/console 2>&l
pf::powerfail:/etc/powerfail
    1>/dev/console 2>&l
dl::once:/etc/some_daemon # typical daemon
co::respawn:/etc/getty console 9600
t0::respawn:/etc/getty tty0 9600
t1::respawn:/etc/getty -t 60 tty1m md_1200 # modem
t2::respawn:/etc/getty -t 60 tty2m md_1200 # modem
t3::off:/etc/getty tty3 co_9600 # ti810 printer
t4::respawn:/etc/getty tty4 9600
t5::respawn:/etc/getty tty5 co_9600
t6::respawn:/etc/getty tty6 4800
p0::respawn:/etc/getty ttyT0 co_9600 # Ethernet 0
p1::respawn:/etc/getty ttyT1 co_9600 # Ethernet 1
p2::respawn:/etc/getty ttyT2 co_9600 # Ethernet 2
p3::respawn:/etc/getty ttyT3 co_9600 # Ethernet 3
```

Figure 6. /etc/inittab file (typical)

This example is for a system with eight serial ports (including two modems and one printer) and Ethernet connections.

Each line controls one process, depending on the init "level" of the system. Level "s" is the single user state which is entered at boot time according to the first line of inittab. When the multi-user state is selected (init level 2), the other lines in this file are used. It would be possible to use different init levels to energize different system configurations. There is more discussion about "init" and "inittab" in sections {10.10} and {10.11}.

You should always have a "getty" running on your unused ports. This will permit you to login on another line to kill a stuck process.

There are more inittab examples in section {16.6}. Refer to "inittab" in section four of the UNIX User's Manual for more details. See the caution "NOTE" on the next page.

6.3 /etc/gettydefs

The `gettydefs` file is used by `/etc/getty` to control terminal characteristics during the login phase.

```

9600# B9600 # B9600 SANE TAB3
#\r\n\nHeurikon System V\r\nlogin: #9600

4800# B4800 # B4800 SANE TAB3
#\r\n\nHeurikon System V\r\nlogin: #4800

md_1200# B1200 # B1200 CS8 CREAD IGNPAR ISTRIP
ICRNL IXON IXANY ISIG ICANON ECHO ECHOE ECHOK
OPOST ONLCR TAB3 #\r\n\nHeu V\r\nlogin: #md_300

md_300# B300 # B300 CS8 CREAD IGNPAR ISTRIP ICRNL
IXON IXANY ISIG ICANON ECHO ECHOE ECHOK OPOST
ONLCR TAB3 #\r\n\nHeu V\r\nlogin: #md_1200

co_9600# B9600 # B9600 SANE TAB3
#\r\n\nHeurikon System V\r\nlogin: #co_1200

co_1200# B1200 # B1200 SANE TAB3
#\r\n\nHeurikon System V\r\nlogin: #co_9600

```

Figure 7. `/etc/gettydefs` file (typical)

Each entry in this file has a name (e.g., "md_1200") which is used as an argument to `getty` in the `/etc/inittab` file. The "#" symbol is a field separator. This file also specifies the "login banner" which identifies the system on the terminal.

The last field of an `/etc/inittab` "getty" line and the first field of a `/etc/gettydefs` line correspond to each other. Those fields do not, in themselves, specify a baud rate. The baud rate is selected in the second and third fields of the referenced `gettydefs` entry (e.g., by "B9600").

In the above listing, the format of the file has been changed slightly for clarity (newlines have been added). Refer to "gettydefs" in section four of the UNIX User's Manual for more details.

See section {16.11} for the definition of "SANE".

- ♣ NOTE: Do not attempt to change the `gettydefs` or `inittab` files unless you are very familiar with their functions and the operation of the UNIX editors. You could easily cause improper system operation if these files are changed in an incorrect manner. See section {10.11} for more discussion.

6.4 /etc/rc

The `/etc/rc` file is a shell script which is executed by "init" as specified by `/etc/inittab`. It controls the mounting of additional file systems (other Winchesters), cleaning up old temporary files, removing program lock files (which may be left from a previous run) and starting background daemons, such as the line printer scheduler.

```

TZ=CST6CDT
export TZ
if [ ! -f /etc/mnttab ] ; then
    > /etc/mnttab
    /etc/devnm / | grep -v swap | \
    grep -v root | /etc/setmnt
fi
set `who -r`
if [ $7 = 2 ] ; then
    mount /dev/wlb /u
    mount /dev/wla /tmp
    rm -f /usr/adm/acct/nite/lock*
    /usr/lib/ex3.7preserve -
    /etc/update
    echo Starting EXOS Network V3.2
    echo netload...
    /net/netload /net/net ; sleep 2
    echo remshd... ; /net/remshd
    echo ud... ; /net/ud
    echo ftpd... ; /net/ftpd
    echo rwhod... ; /net/rwhod
    # mv /usr/adm/sulog /usr/adm/OLDSulog
    # mv /usr/adm/cronlog /usr/adm/OLDcronlog
    # > /usr/adm/cronlog
    nice -20 /etc/cron
    echo cron started
    rm -f /tmp/*
    rm -f /usr/spool/uucp/LCK*
    echo starting lp scheduler
    rm -f /usr/spool/lp/SCHEDLOCK
    /usr/lib/lpsched
fi
sleep 2 # let daemons take hold

```

Figure 8. `/etc/rc` file (typical)

There are two other scripts of interest, `/etc/brc` and `/etc/bcheckrc`, which are not shown here. `/etc/bcheckrc` is the script which asks you to set the date and run `fsck` when switching to multi-user mode. The `/etc/brc` script removes the mount table, "mnttab". By the way, the designation "rc" stands for "run control".

6.5 C-Shell Login Scripts

These scripts are used routinely by the C-Shell.

6.5.1 /etc/cshrc

The first file read by the C-shell at login time is the system-wide /etc/cshrc file. This file does setup functions which are common for every user.

```
setenv TZ CST6CDT
cat /etc/motd
stty ixon echoe
set mail = /usr/mail/$LOGNAME
if ( -e $mail && ! -z $mail ) then
    echo You have mail.
endif
```

Figure 9. /etc/cshrc file (typical)

6.5.2 \$HOME/.cshrc

After the /etc/cshrc is executed, the login process executes the commands in the cshrc file in your home directory. Your "home" directory is specified in the /etc/passwd file. The /\$HOME/.cshrc file contains commands which are specific for each user. This is where you have the most flexibility in configuring the system for your personal tastes. In the following example, the csh aliases provide short names for long commands, allow sloppy typing ("car" = "cat") or to automatically invoke commands with particular options.

```
if ($?prompt) then
# interactive shell
set history = 50
set cdpath = ~
set ignoreeof
set mail = /usr/mail/$LOGNAME
set path = (/bin /usr/bin /etc . $HOME/bin)
set prompt = "%\![el] "
set noclobber
alias from "grep ^From $mail"
alias games cd /usr/games
alias h history
alias ht "history | tail"
alias car cat
alias ls ls7 -F
alias prn pr -nf160
alias vid vi dead.letter
alias vic vi $HOME/calendar
alias who who -uT
endif
```

Figure 10. .cshrc file (typical)

Your `.cshrc` file is also read each time one of your processes starts a new shell, as would be the case when you execute a shell command from within the editor or switch users via "su". Thus, you should not put too much in this script, in order not to slow response times each time a new shell is started. If you have a bunch of commands you occasionally want to execute, it is easy to assign an alias which will "source" the appropriate file. Another way of speeding up response times is to use an "if" statement to skip over the `.cshrc` commands in certain cases, as is done in the above example.

6.5.3 \$HOME/.login

The `.login` file in your home directory is read, once, each time you log into the system. Its function is to configure the system for your particular type of terminal and initialize your terminal, if necessary. It is read after your `.cshrc` file.

```
set noglob
set term = (`tset -e -Q -S -r 925`)
setenv TERM "$term[1]"
setenv TERMCAP "$term[2]"
unset term noglob
setenv EXINIT "set autoindent"
stty -parenb cs8 echoe ixany
news -s
```

Figure 11. `.login` file (typical)

The second line of this `.login` file selects the `/etc/termcap` entry for your terminal. This example automatically selects a Televideo 925.

Some people like to put the line `~/usr/games/fortune` in their `.login` file.

6.5.4 \$HOME/.logout

Your `.logout` script is executed when you type "logout" or "exit" to your C-Shell. It may be used to clean up certain directories or execute programs as you log off the system. Typical entries might be:

```
/bin/rm $HOME/tmp/* (to clean your tmp directory)
clear (to clear your display)
kill 0 (to stop your background pgms)
```

6.6 Bourne-Shell Login Scripts

Since the Bourne shell isn't used as often as the csh, we won't go into detail here. However, two of the typical scripts used by "sh" are shown below for your reference. These scripts perform functions similar to the C-Shell cshrc files, described above.

6.6.1 /etc/profile

This is the system-wide configuration script for all users who use the Bourne shell, "sh".

```

trap "" 1 2 3
export TZ LOGNAME
readonly LOGNAME
cat /etc/motd
TZ=CST6CDT
case "$0" in
-sh | -rsh)
    if mail -e
    then echo "you have mail"
    fi
    if [ $LOGNAME != root ]
    then
        news -n
    fi
    ;;
-su)
    :
    ;;
esac
trap 1 2 3

```

Figure 12. /etc/profile file (typical)

6.6.2 \$HOME/.profile

This script is the personal "sh" configuration file. There would be a separate one for each user.

```

stty echoe erase ^H
SHELL=/bin/sh
export SHELL
PATH=/bin:/usr/bin:/etc::
export PATH

```

Figure 13. .profile file (typical)

6.7 /etc/termcap and terminfo

The UNIX system will operate with a large variety of CRT terminals. The `/etc/termcap` file and, in System V.2, the `/usr/lib/terminfo` files, detail the terminal capabilities for programs which need to know such things as how to clear or scroll the screen. The termcap file is read by the "tset" command in your `.login` script, which loads the TERM and TERMCAP entries in the environment. (The "environment" is a group of variables which "follow" you and any child processes around.) The visual editor, "vi", uses the TERM entry to get terminfo data. The files supplied with the UNIX system cover a wide variety of terminals (over 200). You may add or change the entries, if you wish. Refer to `termcap(5)` and `terminfo(4)` in the UNIX User's Manual for details. If you place your most-used entries near the top of the `/etc/termcap` file, you will find it slightly faster to log in.

```
v4|tvi9504p|9504p|televideo950 w/4 pages:\
:is=\EDF\EC\Ed\EGO\Eg\Er\EO\E\047\E(\EZ\Ew\EX\Ee ^O\
\Ek\E016\E004\Ex0\200\200\Ex1\200\200\Ex2\200\200\
\Ex3\200\200\Ex4\r\200\E\3\E-07 \E3\
:te=\E\3\E-07 :ti=\E\1\E-07 :tc=tvi950:
```

Figure 14. `/etc/termcap` file - portion

6.8 The Clock Daemon

6.8.1 /usr/lib/crontab

The cron table is the control file for the clock daemon, "cron". Cron monitors this file (it checks the file once each minute) and performs any functions specified. For example, certain processes are scheduled to run every day. In System V.2, each user can have their own crontab. The "at" program is also controlled by cron, via "atrun". Lines beginning with "#" are comments.

```
# 0-60min 0-23hour 1-31date 1-12month 0-6day Things-to-do
0,10,20,30,40,50 * * * * /usr/lib/atrun
59 23 * * 6 mv /usr/adm/cronlog /usr/adm/ocronlog ;\
    touch /usr/adm/cronlog
58 23 * * * mv /usr/adm/sulog /usr/adm/OLDSulog ;\
    touch /usr/adm/sulog
58 23 * * * mv /etc/wtmp /etc/owtmp ;\
    touch /etc/wtmp
0 5 * * * calendar -
17,39 * * * * /usr/lib/uucp/uucp.hourly
2 4 * * * /usr/lib/uucp/uucp.daily
6 4 * * 0 /usr/lib/uucp/uucp.weekly
```

Figure 15. crontab file (typical)

6.8.2 /usr/adm/cronlog

Whenever cron does something, it records critical information about what it did along with any error messages in a log file. Here is a portion of such a file:

```
Sat Aug 27 05:00:00

/usr/lib/atrun
calendar -
/usr/lib/atrun
/usr/lib/uucp/uucp.hourly
/usr/lib/atrun
/usr/lib/atrun
/usr/lib/uucp/uucp.hourly
/usr/lib/atrun
```

Figure 16. /usr/adm/cronlog file (typical)

6.8.3 \$HOME/calendar

In most installations, the "calendar" program is automatically run early each morning by "cron". If you have any important events which you would like to be reminded of, enter the critical information in your calendar file. You will receive mail a day in advance of the event. Refer to calendar(1).

```
Aug 30 3:00pm medical
Aug 30 Canadian reps here for visit
Aug 31 11am Meet w/Accountants
Sept 1 12noon Lunch w/Bill
Sept 5 Register for seminar
Sept 12 Bob G to Chicago
Sept 13 Call NY Reps
Nov 29 10:30am meeting w/Mktng
```

Figure 17. calendar file (typical)

There is an improvement which you can make to your /bin/calendar program to alert you to regular weekly events. If you add this code just before the "case" statement in the calendar script, your \$HOME/calendar file can have lines of the form "~Wed 11:30 Staff Meeting".

```

day=`date`
case $day in
Mon*)   echo "`[Mm]on" >>${_tmp} ;;
Tue*)   echo "`[Tt]ue" >>${_tmp} ;;
Wed*)   echo "`[Ww]ed" >>${_tmp} ;;
Thu*)   echo "`[Tt]hu" >>${_tmp} ;;
Fri*)   echo "`[Ff]ri" >>${_tmp} ;;
Sat*)   echo "`[Ss]at|[Ss]un|[Mm]on" >>${_tmp} ;;
Sun*)   echo "`[Ss]un|[Mm]on" >>${_tmp} ;;
esac

```

Figure 18. /bin/calendar ~Day enhancement

6.9 News

The "news" program is usually called from your .login script. Any new entries in the /usr/news directory will create a message when you log onto the system. By typing "news" you can see the new news entries. A special file in your home directory records which items you've seen, so you won't be bothered with "old" news each time you login.

You can create news items yourself by placing a text file in the /usr/news directory. Typical items would be announcements of new computer facilities, e.g., "modem now connected to port tty3", or company events, e.g., "The baseball team is in first place!" Give the files names which relate to their contents, e.g., "modem".

Another way of issuing "news" is to send messages via the "mail" facility. You could compose an announcement and send it to everybody, via:

```
mail jeff dan pete lan debbie dave bruce bob < message
```

The UNIX "news" program should not be confused with the worldwide user's network, "Usenet". Usenet news software (which is not part of UNIX) works with uucp and manages thousands of news articles per week in hundreds of "newsgroups" (technical and non-technical topics). Usenet is an informal network of over 1300 machines, running UNIX and other operating systems, linked by modems, telephone lines and high speed networks. The Usenet population is mostly individuals at corporations in the computer business and universities. Some estimates place the number of people who participate in the network at over 20,000.

6.10 /etc/motd

The "Message of the Day" file is displayed on the user's terminal at login time (by /etc/cshrc or /etc/profile). Unlike the "news" files, this file is output each time a user logs in, whether or not he has already seen it. It could be used to inform users of special system conditions such as: "The printer will be off-line all day Friday". If there are no messages, this file should simply be empty.

6.11 /dev

The /dev directory contains entries which describe the I/O devices, such as serial ports and disk drives. This file provides the link between the operating system and the environment. There is no data stored in this directory, only pointers to device drivers.

cent	rf1	tty	tty3m
console	rf5dd	tty0	tty4
f0	rf5sd	tty1	tty4a
f1	rf8dd	ttyla	tty4b
f5dd	rf8sd	tty1b	tty5
f5sd	rw0b	tty2	tty6
f8dd	rw0c	tty2a	w0b
f8sd	rw0h	tty2b	w0c
kmem	rwlh	tty2m	w0h
lp	sw0h	tty3	w1b
mem	swlh	tty3a	w1c
null	swap	tty3b	w1h
rf0			

Figure 19. /dev directory (typical)

Refer to section {17} for more information on device numbering and naming conventions.

7. THE SHELL

A shell is a program that interacts with the user to allow commands to be executed. The commands usually come from the user's keyboard; however, they could just as well come from a file which has been previously set up with a desired command sequence. An "executable" file in UNIX could be either a true binary file, such as would be produced by compiling a "C" program, or an ASCII text "script" which is a (human readable) sequence of commands that can also be executed by the shell. There are two shells in the Heurikon UNIX system: the Bourne Shell and the C-Shell. The Bourne Shell ("sh") was developed by Bell Laboratories as part of the original UNIX. The C-Shell ("csh") is the result of enhancements made at Berkeley by William Joy, and is usually more popular for interactive use on those systems which have it because of the "alias" and "history" features.

7.1 Csh Alias Feature

Alias substitutions allow you to rename another command or assign a short name to a frequently used or complicated sequence. Example:

```
alias lss /bin/ls -lt
lss
```

Here are some advanced alias examples. The last one, in particular, is helpful if you have ever wanted to "unremove" a file after doing an "rm" command. It causes the "rm" command to save everything you delete. A `~/bin/rm $HOME/tmp/*` command, executed manually or by your `~/logout` file, will really purge the data.

```
alias vim vi /etc/motd
alias look egrep \^!\^ /usr/dict/words
alias append ^cat \!:1 >> \!:2 ; rm \!:1
alias bak cp \!\^ \!\^.bak
alias le ^echo \!-l:q >! /tmp/### ; vi /tmp/### ; \
    cat /tmp/### ; source /tmp/### ; rm /tmp/###
alias rm mv \!\* $HOME/tmp
```

There are other alias examples in the `.cshrc` file, shown earlier in section {6.5.2}. Refer to the UNIX User's Manual and User Guide for details.

7.2 Csh History Feature

The C Shell history feature records your previous commands and allows you to re-execute them without having to retype the whole thing. Example: Type "!!" (called "Bang Bang") to repeat the previous command. Also, if you've ever tried to execute a long command which has a typo in it, you'll appreciate this feature because the typo is easily corrected without retyping everything. There is an example in section {8.20}. The history mechanism also allows you to re-execute a previous command by giving its line number or the first few unique characters of the command. For

example, if you have recently done a `ls -l /usr/lib/uucp > temp` command, then the whole thing can be done again by entering only `!ls`. To have the current command number displayed as part of your prompt, include a line like this in your `.cshrc` file:

```
set prompt = "\! % "
```

For complete details and more examples, refer to the "csh" pages in the UNIX User's Manual and User Guide.

7.3 Wild Cards and Expansions

Wild cards provide a shorthand method of specifying strings or filenames. The shell will "expand" an argument which has wild cards in it before running the program.

<u>Wild card</u>	<u>Description</u>
?	Stands for any single character.
*	Stands for any number of any characters.
[]	Allows a group or range of characters
{}	Allows variations in strings

For more information, refer to the CSH pages in the UNIX User's Manual and the UNIX User Guide. The curly braces, "{}", are the most fun.

```
echo /usr/{fred,joe}/{rc,xx} will give:
/usr/fred/rc /usr/fred/xx /usr/joe/rc /usr/joe/xx
```

7.4 Shell Scripts

You have already seen examples of shell scripts. The `.cshrc`, `.login` and `/etc/rc` files, shown earlier, are all shell scripts. The two shells have slightly different formats for the built-in commands; so, be careful to keep them straight. The Bourne shell ("sh") will only run scripts written for it. The C-Shell ("csh") will accept either. If you feed a Bourne Shell script into a C-Shell, the csh will simply start a "sh" to execute it. The very first character of the script is used by the C-Shell to figure out what type of script it is. If the first character is a "#", which signifies a comment line, it is a C-Shell script. If there isn't a "#" at the beginning, the script is for the Bourne shell.

Although "csh" is more popular as a command interpreter (if available), most scripts are written in "sh" format. This is due to historical reasons, and to maintain portability of programs between UNIX systems. It is generally accepted that the command language of "sh" is better suited for most scripts.

To install a script, follow these steps:

- [1] Use the editor ("vi") to create a file and write the script. Usually this file will be in one of your directories during development.
- [2] Test the script by entering `source scriptname` (for a csh script) or `sh scriptname` (for a sh script).
- [3] After it's debugged, use "chmod" to make it executable, and put the file in /usr/bin or one of your own directories.
- [4] Execute a `rehash` command. If you put the file in an unusual place, be sure your "path" variable is set properly, so your shell can find it.

Frequently, tasks can be implemented with a script which otherwise would require many hours of coding and testing using a conventional program. Some UNIX commands are actually shell scripts instead of binary programs. For example, the following commands are shell scripts:

```

/bin/spell      /bin/calendar
/bin/diff3     /bin/man

```

Some of these scripts call binary programs to do the really hard work. The scripts put things in logical order and allow new commands to be built on the old ones.

You can use the "file" command to find other command scripts and the "cat" command to print them out.

The following sh shell script, "whereis", will tell you where a program is located.

```

: # force /bin/sh
# whereis: Look in reasonable places for commands.
places="/bin /usr/bin /usr/local/bin \
/etc /usr/lib . $HOME/bin /lib"
for param do
  echo $param: \\c
  for place in $places ; do
    if [ -x $place/$param ] ; then
      echo $place/$param \\c
    fi
  done
done
echo
done

```

Figure 20. `whereis` script

The standard "mail" program does not allow interactive message editing. This is a shell script which can be used to compose and send mail using "vi".

```

: # vi mail
PATH=/bin:/usr/bin
t=/tmp/vmail$$
if test $# -eq 0 ; then
    echo "To:" \\c
    echo "To:" "`line`" > $t
    echo "Subject:" \\c
    echo "Subject:" "`line`" "\ >> $t
    echo Fetching vi... Go ahead with \^o\^ and text entry
else
    trap "rm -f $t ; exit" 1 2
    echo "Subject:" \\c
    echo "To: $* \nSubject:" "`line`" "\n" >> $t
    echo Fetching vi... Go ahead with \^o\^ and text entry
fi
trap "mv $t dead.letter ; exit" 1
trap "" 2
vi +3 $t ; # get the text of the msg
trap "echo Saving mail in dead.letter ;\
    mv $t dead.letter ; exit" 2
echo "Okay to send (y/n)? " \\c
response="`line` "
if test $response -a $response != y ; then
    echo "Saving in dead.letter"
    mv $t dead.letter
    exit
fi
echo ...mailing
(
    set `nice sed -n -e "/^$/q" -e "s/^To:.\(.*\)$/\1/p" \
        -e "s/^Cc:.\(.*\)$/\1/p" $t` "" # get "To:" & "Cc:"
    if test $1 = "" ; then
        echo "To: line error, saving in dead.letter"
        mv $t dead.letter
        exit
    fi
    for i do
        nice -20 /bin/mail $i < $t
    done
    rm -f $t
)&

```

Figure 21. 'vmail' script

7.5 Special Characters

Some characters have special significance to the kernel or the shell. In the following table, the notation "[^]X" means control character "X". Some of these characters may be changed by using the "stty" command.

<u>Character</u>	<u>Function</u>
[^] S	Stop output
[^] Q	Restart stopped output
[^] D	EOF (end of file)
@	Line kill (delete line)
#	Gridlet, Line kill (alternate) (also csh script identifier)
DEL	Interrupt
[^]	Quit (dumps core)
[^] \	Quit (dumps core)
[^] H	Backspace (erase)
[^] I	Tab
'	(single quote)
"	(double quote)
`	(back quote)
\$	(dollar sign)
*	wild card, any group of characters
?	wild card, any single character
!	("bang"), csh history feature
[^]	(caret), csh history feature
&	(ampersand) background
~	(tilde)
()[]{}<>	(dodads) used by the shell
\	escape

Table 4. Special Characters - partial list

You must be careful when using any of these characters. For example, the commands:

```
echo Continue?
echo It cost $10.00.
mail sysb!fred
```

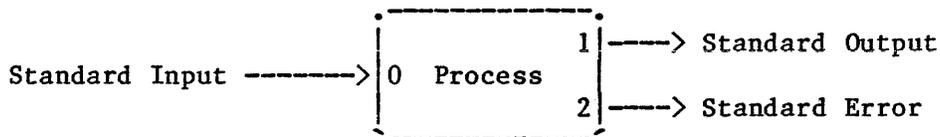
will not work, but these will:

```
echo "Continue?"
echo `It cost $10.00.`
mail sysb\!fred
```

Refer to stty(1), csh(1), sh(1) and termio(7) in the UNIX User's Manual for more information.

7.6 Redirection of I/O and Pipes

In UNIX, a program has files called the "standard input", "standard output" and "standard error" associated with it. The shell sets these up to point to your terminal under normal circumstances so that you can communicate interactively with a program. The standard input of your login shell, for example, comes from your terminal keyboard, and the standard output and standard error go to your CRT display. When the shell starts a command up for you, it usually connects your terminal to the standard input, standard output and standard error of the command.



When you issue a command to the shell, however, you can specify that the standard input, output and/or standard error connect somewhere else. This is a very fundamental and useful feature of UNIX. It allows you to control the input and output of a command. Also, it allows you to logically connect a series of commands together to perform a more complicated function than could any single command standing by itself. For example, here are some examples of "redirecting" the I/O:

- ♣ Redirecting input. Assume you have a lengthy mail message to be sent to someone. You can compose it in advance using "vi". Say it is named "textfile". When you are ready to send it, all you have to do is enter:

```
mail fred < textfile
```

and off it goes. The "<" symbol redirects the standard input of the mail command to be from "textfile" instead of your keyboard.

- ♣ Redirecting output. The standard output of a command can be sent to a file. This means that you can save the output of a command or have the output sent to a particular I/O device.

```
ls -l /usr/bin > ls.output
```

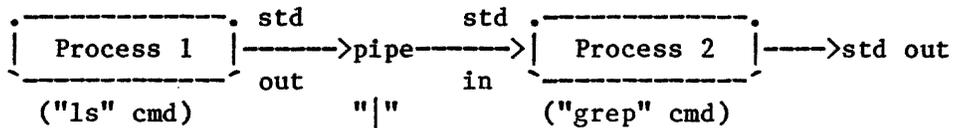
saves the output of the "ls" command in a file called "ls.output" in your current directory. You could also use a ">>" symbol, in which case the output would be appended to the file. A command such as 'date >/dev/console' will print the output (e.g., date and time) on the console port, even if you're executing the command at some other terminal.

- ♣ Pipes. This feature combines the redirection of output with the redirection of input, using a "|" symbol. For example, you can take the output of a "ls -l" command and have it fed to the input of

another command, "grep", as in:

```
ls -l /usr/lib | grep "^drwxrwxrwx"
```

The output of that series of commands will be a list of any subdirectory of /usr/lib having permissions 777. (You could use the "find" command to do this, too.)



Pipes (also known as FIFOs) have a wide application in UNIX. It is not uncommon to find a whole series of commands linked together using "|". A text file can be processed and manipulated in very intricate ways using a sequence of simple UNIX commands. The commands in a pipeline are frequently referred to as "filters". Here is another example:

```
cat f1 f2 - | tr "[a-z]" "[A-Z]" | sort | head -20 > f.out
```

That sequence concatenates files "f1", "f2" and whatever you type on the keyboard up to a Control-D, converts all lower case letters to upper case, sorts the lines alphabetically and stuffs only the first 20 lines of the result into "f.out".

- ◆ Redirecting the standard error. Perhaps you have a large program to compile, and you expect some error messages which you want to save. Normally, the messages will just come to your terminal. However, you can redirect the output of the compile command and put it into a file, as follows:

```
cc -v test.c >& cc.errors
```

In this case, both the standard output and the standard error are redirected to the "cc.errors" file by the ">&" symbol. The standard output of the "cc" command would be information from the compiler, e.g., the list of compile steps, and the standard error would be the compilation error messages.

The standard output and standard error may be directed to different files by using one of these forms, depending on the shell in use:

```
for csh:  ( cmd > stdout ) >& stderr
for sh:   cmd > stdout 2> stderr
```

7.7 Background Commands

A "background" task is one which is running while you are still able to issue new commands. For example, in the previous section, we showed how to redirect the output of the "C" compiler. By running that command in the background, you will be able to run other commands while the compiler is still working. You won't have to wait for it to finish first. That is done by adding a "&" symbol after the command, as follows:

```
cc -v test.c >& cc.errors &
```

If you didn't redirect the output (including stderr), any error messages would appear on your terminal, even if you were executing another program at the time - a condition you may or may not want.

Another reason for running a command in the background is that it may take considerable time to run, and you may want to logout before the command finishes. If you run it in the background, it will continue to run even after you have logged out. Here is an example:

```
make bigpgm > make.out &  
logout
```

Most background tasks do not need to be run at normal priority, since you probably are not waiting for the results. Thus, you can be "nice" to other users by lowering the priority of your background job, as in:

```
nice make bigpgm > make.out &
```

You can check on the status of your background tasks two ways.

- ♣ Use the "ps" command to see if they are still running.
- ♣ Enter `^wait^`. If your tasks have finished, you will get a prompt. Otherwise, the "wait" command will not return until they have finished. You can hit DEL to a hung "wait" command to get a prompt back. If you do that, wait will list your background tasks which are still running.

When you start a background task, the shell prints the task's process number. To stop a background task, you can enter `^kill pid^` where "pid" is the process number of the background task you want to stop. You can also enter `^kill 0^`, which will kill all your background tasks.

Some background tasks may not be willing to die easily. They may be "ignoring" the termination signal or they may be waiting for a device driver to complete an I/O command. Programs which are simply ignoring the termination signal can be convinced you mean business by using `^kill -9 pid^`. Programs which are waiting for I/O (such as character output) will not die until the I/O command is completed. For more information on this later case with respect to the serial ports, refer to section {3.5}.

By the way, a C program can put itself into the background by executing a `fork(2)` system call, such as:

```
if ( fork() != 0 ) /* if parent or error */
    exit(0); /* parent returns, child continues */
```

7.8 Use of Shell Variables

Variables may be defined by either "sh" or "csh" and used for that shell or "exported" to the environment for use by other shells and programs. Certain variables have special meanings to the shell. For complete details, see the appropriate shell manual pages. Here is a summary and some hints for using shell variables.

cdpath When you issue a "cd" or "chdir" command, this variable is used as an list of alternate search paths if the specified directory is not found.

mail Specifies what file or files you want checked for mail and how often the check is to be made. Thus, you can have your csh check for new mail every few minutes, rather than the 10 minute default, and you can have more than one file monitored for new mail.

history Specifies how many "old" commands to keep around.

path Specifies where to look for commands.

prompt This variable is used by the csh as a command prompt. It can be set to any string you like, and can set to display the command number, as in the example below.

Variables may be set from the command line or in a script, such as your `.cshrc` file, as follows:

```
set prompt = "\! What now, master? "
set history = 50
set cdpath = (~ /usr)
set path = (/bin /usr/bin /u/usr/bin /etc $HOME/bin . /usr/bin/uc)
set mail = (90 /usr/mail/$LOGNAME /usr/mail/root /usr/mail/adm)
```

To print a list of the current variables, enter `set` without any arguments. To list those in the environment, enter `env`. You may want to pipe the output through `see` if any of the variables contain non-ascii characters.

Shell variables find their biggest use in scripts as containers for character strings and loop counts.

7.9 Shell Layering

(System V.2 only)

Shell layering allows you to have multiple interactive commands running concurrently. You can switch from one command to another without losing input or output. This is accomplished by logically connecting your communication port to only one shell at a time. The `shl(1)` command controls this feature. For more details, refer to `shl(1)` and the description of `cs(1)` in the User Guide.

Here are a couple things which aren't obvious when reading the manual pages:

- [1] The "commands" listed in the `shl(1)` pages may be abbreviated. For example, the "resume" command can be entered simply as `^r^`.
- [2] To have shell layering automatically start when you begin a session, put `^shl^` as the last line in your `.login` file. This will cause the layering manager to start when you log in. Then, manually enter `^c^` or `^c name^` to start your first layer.

8. INTERESTING COMMANDS

Here are some commands you may try. These are all fairly harmless, so don't worry about causing any problems. Some commands are built into the Shell, but most live in /bin or /usr/bin. If you want to obtain more information about these commands, look in section one of the UNIX User's Manual under the appropriate command and, since some of the commands are built into the shell, in the CSH(1) pages of that manual.

8.1 Process Status

```
ps -e          Display process status, all users
ps -ef        Display process status, "full" form
ps -el        Display process status, "long" form
```

8.2 Change Directory and List

```
cd            Go to home directory
ls           List contents
/bin/ls      Force Bell Labs ls
/usr/bin/ls7 Force Berkeley ls
ls -a        Show the "." files, too
cd /etc      Change to the /etc/directory
ls -l        Display contents in long form
cd /bin      Change to /bin directory
ls m*        Display all commands which start with "m"
```

8.3 Change Directory and Print Working Directory

```
pwd          Display current directory
cd           Change to HOME directory
pwd          Display current directory
```

8.4 Pattern Search

```
grep rootcsh /etc/passwd
grep tv925 /etc/termcap
grep "^pre" /usr/dict/words
grep oo /usr/dict/words
grep tset ~/.login
```

8.5 Display File Contents

```
cat /etc/bcheckrc Used for ASCII text files
see /etc/utmp     Used for ASCII or non-ASCII files
more /etc/termcap For long files
```

8.6 Display Who is Logged On

```
who -uT      Report who is logged in
last         Report login history
```

8.7 Sleep

```
sleep 10 ; echo wakeup
```

8.8 Display Environment and Shell Variables

```
set          Display shell variables
env          Display environment
```

If your environmental variables contain control characters, you may want to pipe the output of "env" through the "see" program:

```
env | see    Display environment
```

8.9 Date and Time

```
date        Display current date and time
```

8.10 Translate Characters

```
tr "abcdefghi" "ABCDEFGHI"
This is a test
```

(Type a Control-D when done)

8.11 Copy, Move and Remove

```
cd
cp /etc/passwd localpasswd
ls
mv localpasswd tempxx
ls
rm tempxx
```

8.12 Display File Types

```
file *
file /etc/*
file /usr/bin/*
```

8.13 Check Spelling

```
spell textfile
```

8.14 Echo Arguments

```
echo this is a test
echo test *
echo "test *"
echo /bin/*
echo $mail      Echo the contents of a shell variable
echo $path
echo $LOGNAME   Echo the contents of an
                environment variable
```

8.15 Time a Command

```
time ps -e      Time the "ps -e" command
```

8.16 Send Mail

```
mail fred
Don't forget the meeting at 3pm, RE: Berk Contract.
.
```

8.17 Display Terminal Options

```
stty -a
```

8.18 Repeat the Previous Command

```
!!
!fi
```

8.19 Display History and Aliases

```
history
alias
```

8.20 Correcting Simple Errors

```
grwp guest /etc/passwd
^w^e
```

8.21 Repeating a Command

```
repeat 10 date
```

8.22 Head and Tail

```
head -3 /etc/passwd
tail -2 /etc/passwd
tail -f file          Display file as it grows
```

9. OTHER THINGS TO LEARN (TOOLS)9.1 Summary

Here is an outline of the important items which you should learn about UNIX before you can graduate from the novice category:

♣ Familiarity with the following commands:

ps	ls	cd	pwd	cat	head
tail	more	grep	mv	rm	date
who	mail	wc	pr	dc	echo

♣ Use of the csh history ("!", "^") and aliasing features.

♣ Familiarity with the visual editor, "vi".

♣ Understanding of a "regular expression" as defined in the "ed" manual pages in the UNIX User's Manual. A regular expression is used to specify a set of character strings which meet particular requirements and is widely used by the UNIX editors and pattern matching programs such as "grep".

♣ Familiarity with the C language and the C compiler, "cc".

♣ Familiarity with other commands:

diff	sort	spell	cut	fsck	tar
cpio	at	make	split	strings	stty
tee	tr	sed	awk		

♣ Familiarity with the text formatter, "nroff", and the "mm" macros.

If you play with the UNIX commands, you will find they become more and more useful as you recognize how they can be applied to everyday needs. Think of them as "tools" which you can use as the need arises.

9.2 Move vs. Copy vs Link

Sometimes, there is confusion regarding the differences among these three commands:

- ♣ Move ("mv") will pick up a file and move it to another directory or allow you to change its name in the same directory. The actual data stored on disk is not touched in any way, only the directory pointers (links) are adjusted. The file attributes (owner, size, permissions, etc.) are not changed. However, if you try to move a file between partitions or drives, the move command will automatically be turned into a copy.
- ♣ Copy ("cp") will duplicate the file in another directory or duplicate it in the same directory, using a different name. This command clones the file.
- ♣ Link ("ln") will connect, or "link", a file into two (or more) places in the file system. The actual data will not be duplicated; however, the file can be referenced by any of its names. If one of the files is modified, the change will instantly be seen in all the other files because, in fact, they are identical. The file attributes will be the same for each link. You can use the "ls -l" command to find linked files.

Links are used frequently when two or more programs are so similar that they may as well be one program. The "ex", "vi" and "edit" programs are links for that reason. When the program executes, it can easily find out which name was used to call it, so that it is able to do the correct function.

Incidentally, the "mv", "cp" and "ln" programs are all links to the same program. Why? Because they do tasks, which are so similar in nature, that it is simpler to have just one program which can do all three.

9.3 The Visual Editor

Although there are many commands associated with "vi", only a few are needed to get going. Be sure to read about vi in the UNIX User's Guide.

arrows	move cursor (alternates: h,j,k & l)
^d (^u)	move down (up) 1/2 screen
^f (^b)	move forward (back) full screen
^	move to start of line
\$	move to end of line
G	move to end of file
nG	move to line "n"
dw	delete word(s)
dd	delete line(s)
<ESC>	escape from insert or append mode
cw	change word(s)
D	delete to end of line
C	change to end of line
i	insert characters
I	insert characters (beginning of line)
a	append characters
A	append characters (end of line)
s	substitute characters
rx	replace i character (with "x")
R	replace characters
x	delete character(s) (at cursor)
X	delete character(s) (before cursor)
p	put after
P	put before
Y	yank (copy) line(s)
xp	transpose characters
.	repeat previous command
o	open new line (below)
O	open new line (above)
u	undo last change
U	undo all changes on this line
^R or ^L	redraw screen
:map	define keystroke macro
:r fl	read file in
:r !cmd	read in output of cmd
:w fl	write file out
:e fl	change file
:n	edit next file
:q	quit
ZZ	write and quit (same as :x or :wq)
:!cmd	shell escape
%	show matching (), [], or {}

Table 5. ^vi^ command summary

The "ex" and "vi" UNIX editors have a number of interesting features.

- ♣ You can abbreviate keystrokes by using the "abbr" function.
- ♣ Certain options can be adjusted via the ":set" command.
- ♣ A directory full of related source files can be cross-referenced using the "ctags" program. Then, the editor's ":ta" command will let you move around between "C" routines, as if they were all part of one monster file. You don't have to remember where each routine is. UNIX makes it easy to be lazy.
- ♣ The "map" command allows you to customize your keyboard and have certain keys take on special functions. Often used keystroke sequences can be crunched into a single key by using the "map" command. For example, you can program a key which will write your updated file to disk and then feed it to the compiler. There are more examples below.
- ♣ You can set the "EXINIT" variable in your environment via your \$HOME/.login file to initialize "vi" with the features, options and commands which you most prefer when you use the editor.

Here's an advanced example. The penultimate line, shown below, programs the "K" key to write your file out, feed it to the "spell" program and append any misspelled words to the end of your file. The last line programs the "Q" key to pick out one of those words and find it in the text for you so you can correct it.

```
setenv EXINIT ^set autoindent|map ^C :w^M:\! |\
  map ^X :w^M:\!%^M | map g lG | map z xPP |\
  map V :$r /usr/$HOME/.signature^MO^D-----^[ |\
  map K :w^MGoSpellList^[:$r spell %^M |\
  map Q GI/^A>^["zdd@z^
```

The above example is split into five lines for clarity. It really should be all on one line. The "^" symbol means the following character is a control character. To enter them while using "vi", type control-V followed by the control key. E.g., "^M" is entered by typing control-V then control-M.

- ♣ If you do much "C" programming, you'll love the showmatch and "%" function.
- ♣ Crash Recovery. If the system is brought down while you are editing, or if you're using a telephone link and the connection is broken, you will not lose your changes. To recover all but the last few changes you made, go to the same directory you were in before the crash and enter `^vi -r filename^`.

9.4 Other UNIX Editors

Here is a summary of the various editors which are provided as part of the UNIX operating system.

- ed This is the standard UNIX interactive, line-oriented, text editor. The "ed" manual page in the UNIX User's Manual details the concept of a "regular expression", which may be used in "ed" and numerous other UNIX programs to define sets of character strings which match or conform to certain specifications.
- red "Red" is a restricted version of "ed". It only allows files in the current directory to be edited. "Red" and "ed" are links to the same program.
- ex This is the (Berkeley) enhanced UNIX line editor. It has some features which make it ideal for editing programs, and has a "visual" mode ("vi") which allows full screen editing on CRT terminals. See the previous section for more information.
- edit This is a version of "ex" for beginners. It automatically defines some default parameters and is easier to learn. It is a subset of "ex".
- vi This is the "visual" portion of the "ex" editor. It starts up in visual mode rather than the "open" mode of "ex". You can switch back and forth between open and visual from either "ex" or "vi".
- view "View" is equivalent to "vi", except it operates in read-only mode. You will not be allowed to overwrite a file with the usual write commands. It should be used if you want to examine a file and protect it against accidental modifications. "vi", "view", "ex" and "edit" are all links to the same program.
- sed The "sed" stream editor finds extensive use in UNIX as a filter. It processes a file, line by line, and applies a set of editing commands to each line. It is typically used to reformat files or extract lines which meet certain conditions. For example, the following "sed" command will output only the lines which occur between the strings "HX" and "HE" and change any occurrence of "test" to "text".

```
sed -n -e "s/test/text/" -e "/HX/,/HE/p" file
```

See Appendix {B} and the "vmail" script in section {7.4} for other examples. "Sed" also permits editing files which are too large for the other editors to handle.

9.5 Compiling a 'C' Program

If you are going to write a program in "C", follow these steps:

- [1] Use the editor ("vi", "ex" or "ed") to create the source code file. Use a ".c" suffix on the file, as in "testpgm.c".
- [2] Invoke the compiler and loader by entering `cc -v testpgm.c`.
- [3] Execute the result by entering `a.out`. It may be necessary to enter `./a.out`, if there is another a.out file in the shell's search path.
- [4] You can enter `mv a.out testpgm` to rename the a.out file. Then enter `testpgm` to execute the program.

You can compile the program directly into "testpgm" via

```
cc -v testpgm.c -o testpgm
```

If you move the program to /bin or /usr/bin, don't forget to execute a `rehash` command. Also, you may want to change the permission modes of the file to protect it from other users. Use "chmod", "chown" and "chgrp".

Some programs need special "libraries" to run. For example, there are math libraries and terminal control libraries ("curses", "termcap"). The libraries are in /lib and /usr/lib. The standard "C" library is automatically used by the link edit phase of "cc". Other libraries must be specifically listed if needed, as in:

```
cc -v testpgm.c -o testpgm -lcurses -ltermcap
```

Information on the "curses" library is not included in the standard UNIX manuals. Contact us if you need curses documentation.

9.6 Linking C, FORTRAN, Pascal and Assembler Programs

Object modules created from programs written in different languages can be linked together as long as certain conventions are observed.

Care must be taken when passing subroutine arguments. FORTRAN is "pass by address" while C and Pascal are "pass by value". Pass by address means that subroutine arguments are pointers to a data item, while pass by value means the argument is the actual data. This difference is handled by properly coding all shared subroutines and subroutine calls made by C and Pascal routines.

Another interface problem concerns subroutine and variable names. All compiler variables and routines names have a prepended underscore. In addition, the Green Hills FORTRAN compiler appends an underscore to names.

This means that C and Pascal programs must do so, too.

The following C, FORTRAN and assembler programs illustrate this interface:

```
main() /* This is the main entry point */
{
    int x,y,result;
    f_init(); /* you only need to do this here to allow FORTRAN I/O */
    x = 3; y = 4;
    result = fort_(&x,&y); /* pass addresses to FORTRAN */
    printf("Result = %d\n", result); /* should print "22" */
    exit();
}

cprog_(arg1,arg2) /* this routine is called from the FORTRAN program */
int *arg1,*arg2; /* use FORTRAN pass by address convention */
{
    return( addit(*arg1,*arg2) ); /* gets values and pass to asm pgm */
}
```

Figure 22. Linking Languages - C fragment (c.c)

```
subroutine fort(in1,in2)
integer in1,in2,temp,c1
return( cprog(in1*in2,10) )
end
```

Figure 23. Linking Languages - FORTRAN fragment (fort.f)

```
.text
.even
.globl _addit
_addit: movl    sp@(4),d0      | get x
        addl    sp@(8),d0      | add y
        rts                               | return sum
```

Figure 24. Linking Languages - Assembler fragment (ass.s)

The following command sequences will compile and link these programs:

```
gf77 -c fort.f                gf77 -c fort.f
as addit.s -o addit.o         or gcc c.c addit.s fort.o
gcc c.c addit.o fort.o
```

If you are having problems, ask the compiler to leave behind the ".s" file; then examine it to see what the program is really doing. Information about register allocation can be found in the compiler documentation. By the way, the UniSoft assembler ("as") uses MIT mnemonics if you're System V; Motorola-like if System V.2.

9.7 The Make Command

Typically, a large UNIX program is broken up into numerous, easy to handle, small source files. Each source file contains the routines for some particular and distinct function. For example, the UNIX kernel is composed of about fifty "C" source files, and the device drivers and system configuration files add another twenty. It would be very difficult to manually control all those files and to remember which ones are current, which have been updated and how, in the end, to compile and link them all together.

The "make" facility is a tool for maintaining such a group of programs. One control file, called a "makefile", is used to describe the relationship between the source files and to specify how to compile and link the final result. Then, a single command, "make", will cause the end product to be recreated. Any source files which have been updated since the last "make" will automatically be recompiled and included in the result. "Make" does this by checking the modification times associated with each file, comparing them against the times of the various intermediate and end products and invoking the specified command sequence, if necessary, to bring the intermediate files up to date. Any old "dependent" files are regenerated.

It is also quite common for a number of slightly different "versions" of a program to be required. If one of the source files which is common to all versions is updated, then one command, "make", will automatically recreate all of the variations.

Makefiles can get quite large themselves. Which means, however, there is that much less for the programmer to remember. The following, relatively short, makefile is used to maintain a group of C programs.

In this case, 'make' already understands how to compile a 'C' program; it just has be told how to put the pieces together. Typing 'make' will compile and link any (of the seven) updated C programs to create "hds". Typing 'make tags' will produce a cross reference listing of C routines for the editor.

It is not necessary for the files to be "programs"; any group of related files can be maintained via 'make'. For example, here is a short makefile which is used to maintain a directory of nroff-type documentation files.

```

# description file for HDS terminal window control

INCLUDE      = -I.
SHELL =      /bin/sh

DEFS =
LINTFLAGS =  $(DEFS) $(INCLUDE)
CFLAGS =     $(DEFS) $(INCLUDE) -O -v
LFLAGS =     -s

FILES = bmove.o util.o data.o fkproc.o init.o main.o set.o
LINTFILES = bmove.c util.c data.c fkproc.c init.c main.c set.c

all:  hds

hds:  $(FILES)
      cc $(LFLAGS) -o hds $(FILES)

lint:
      lint $(LINTFLAGS) $(LINTFILES)

tags: /tmp
      ctags *.c > tags

```

Figure 25. Sample makefile (C programs)

This next makefile will automatically feed any updated file with a ".t" suffix to "nroff" and place to output in a corresponding ".f" file. It also makes a backup copy of any modified ".t" file. The UNIX make facility is usually used to maintain directories of "C" programs.

```

# Description file for documentation using nroff

ffiles = intro.f ports.f devices.f baud.f
bfiles = intro.bak ports.bak devices.bak baud.bak

.SUFFIXES:
.SUFFIXES: .f .t .bak

.t.f :
      rm -f $@
      nroff -cm -rB2 $< >$@

.t.bak :
      cp $< $@

all: $(ffiles) $(bfiles)

```

Figure 26. Sample makefile (nroff files)

9.8 nroff/troff

The "nroff" and "troff" programs are document formatters. Nroff is for use with a typewriter-like printer; troff is for a typesetter or laser printer. Both programs allow free form text input, intermixed with formatting commands.

There are a number of macro packages which make it very easy to use the document formatters. A "macro" is a sequence of formatter commands that perform some specific function (such as outputting a page header) but which can be invoked with a simple, one-line command. For example, this guide was produced using the "mm" macros. There are macros for specifying section headers, page headers, page footers, and style. For those of you who are interested, we used the following mm parameters:

```
.SA 1
.ds HF 2 2 2 2 2 2
.TC 1 1 2 0 ...
```

The single-line ".TC" command prints the table of contents and the "Figure" and "Table" lists, which you will find at the beginning of this guide.

We're showing off a little on this page, as you can see. The ".2C" (two column) command is all it takes. Automatic hyphenation has also been used on this page, by using ".nr Hy 1"

In addition to the preprogrammed macros, you can write your own, or modify the existing ones, if you need some special functions. For most situations, however, the existing macro packages will be sufficient.

Here is a summary of some of the programs and macro packages which are provided as part of UNIX:

- mm The "mm" macros are for general purpose documents, such as this UNIX guide. It is the most general and complete set of macros for document preparation.
- mv Macros for making slides and viewgraphs with troff.
- man The UNIX manual pages are prepared using the "man" macros. They are designed to make documentation of UNIX commands and files conform to a standard format.
- mptx Macros for formatting the permuted index.
- tbl A program to format tables with headings, columns and borders. Tbl automatically computes column spacings, etc.
- eqn A program to process mathematical equations for troff.
- neqn Same as eqn, but for use with nroff.

Refer to the UNIX Document Processing Guide for more information concerning the text formatting features of UNIX.

```
* * * * *
*
* If it's square, it's fish; *
* If it's round, it's a burger *
* (Else, it might be chicken.) *
*
* * * * *
```

Here are some more details on how this guide was produced.

- ♣ Level one headings and the top of page headings are produced by the "HH" macro, as follows:

```
.de HH
.OH "^^\ $1 ^"
.EH "^^\ $1 ^"
.SK
.H 1 "\ $1"
..
```

- ♣ The nroff command is

```
nroff -cm -rW74 -rB2 -e -T450-12 guide.t > guide.f
```

- ♣ The index is generated by writing a special line for each item and processing the nroff output with "sort" and a bunch of fancy "sed" and "awk" commands. Then, nroff is used a second time to format the index pages. Appendix {B} contains the macros and scripts used to do this.

- ♣ Some printers cannot backspace or do reverse linefeeds. In that case, the nroff output file may be printed using:

```
cat guide.f | col -x | under | lp -t"UNIX Ref Guide"
```

The "col -x" program process the reverse line feeds used by nroff to generate the .2C format of the previous page or output of the `tbl` macros. "Under" is a short program which processes the backspaces used by nroff to produce bold or underscored text. It converts lines with imbedded backspaces into two lines separated by a carriage return (no line feed). The source code for "under" is shown on the next page. "lp" prints the result.

- ♣ Although the headings and lists are automatically numbered, references within the document are coded by hand. Nroff makes only one pass through the document so there is no way to generate automatic forward references. A macro was used to format the references and place them inside curly braces, mainly to make them easy for "grep" and the editor to find.

```

#include <stdio.h>
main()
{
    char line1[140];
    char line2[140];
    int pflag;
    int c, i,j;

    pflag = 0;
    i = 0;
    while ( (c=getchar() ) != EOF ) {
        line1[i] = c;
        line2[i++] = ' ';
        switch (c) {
            case '\b':
                --i; /* back up to \b */
                line2[i-1] = line1[i-1];
                line1[i-1] = getchar();
                pflag = 1;
                break;
            case '\n':
                --i;
                for (j=0;j<i;j++)
                    putchar(line1[j]);
                if (pflag) {
                    putchar('\15'); /* cr */
                    for (j=0;j<i;j++)
                        putchar(line2[j]);
                    pflag = 0;
                }
                i = 0;
                putchar('\n');
                break;
        }
    }
    exit(0);
}

```

Figure 27. `under` program

9.8.1 nroff Terminal Description Files

Nroff uses the "-T" option to find a description file in /usr/lib/term which contains details about how to control a particular printer (e.g., how to do boldface characters). Numerous description files are provided with the system. For example, "-T450-12" will operate a Qume printer in 12-point mode. Additional documentation is available if you need to write your own description files; contact our Customer Service Department.

9.9 awk

"Awk" can be used to scan and process a text file according to some set of commands. It is similar to the "sed" stream editor in function; however, it allows very complex operations to be performed. It can be programmed to make decisions and do various things, depending on the particular input text. Awk commands resemble "C" language source code.

Awk was a big help in formatting the index section to this guide. For details on how that was done, including an awk program, see Appendix {B}.

Refer to the pages in the UNIX User's Manual and the UNIX Support Tools Guide for more details. It is worth the effort to experiment with this program to get a feel for its capabilities. It may come in handy.

9.10 SCCS

The UNIX "Source Code Control System" is a collection of programs for controlling software versions and updates. Basically, the SCCS software allows you to keep track of numerous program versions which have resulted from modifications. It allows old program versions to be extracted from the sources without having to keep a separate copy of each release. For more information, refer to the UNIX Support Tools Guide.

9.11 Debuggers

9.11.1 adb

"Adb" is the System V.0 UNIX debugger program. You can use it to examine a core dump or to single step through a program, looking at registers as you go.

The command syntax is a little hard to cope with at first, so, to get you started, try this: Compile a program (but do not "strip" it). Leave the output in "a.out". If you are on system V.2, convert your program using `^coffbin -s -u a.out^`. Execute it and hope for a gross error which "dumps core". Enter this sequence to load the program (assumes name is a.out) and the core file:

```
adb
$r
$c
```

This will display a register dump taken at the time of the error, the instruction near the one which caused the error (or one very near to it) and a stack trace. The stack trace will help you determine which subroutines were called and which caused the error. The subroutine argument addresses are also shown, so you can check the actual data passed to the routines.

Other commands to learn are `^?i^` to disassemble code (in a.out) and `^/x^` to display hexadecimal values (in the core file). To "go to" a particular

routine or variable, just enter its name. Now, read the manual.

In system V.2, `adb` and `coffbin` are provided for use by the rebuild procedure. However, they are not supported commands.

9.11.2 sdb

"Sdb" is the System V.2 UNIX Symbolic debugger program. It includes the basic features of `adb` but also allows you to debug a `C` or Fortran program at the source code level; `sdb` understands source code text and can relate variable by name. Look in the System V.2 UNIX Programming Guide for an `sdb` tutorial.

Sdb is not available with system V.0.

10. ADMINISTRATIVE FUNCTIONS

In addition to booting the system and doing backups, procedures that are described elsewhere in this guide, there are other duties for the system administrator.

- ♣ **Caution:** The system administrator must be super-user to perform many of his duties. However, since you could cause accidental damage to your system while super-user, it is unwise to be super-user unless absolutely necessary. You should have a ordinary user login, switch to super-user (via `'su rootcsh'`) only when needed, and `'exit'` back to your regular login as soon as possible.

10.1 Adding New Users - Removing Old Users

To add a new user to the system, become the super-user and do the following:

- [1] Add a new line for the user to the `/etc/passwd` file. Use "ed" or "vi".
 - a. Duplicate the last regular user line.
 - b. Change the logname field of the new line.
 - c. Change the user id field. For regular users, assign a user id number of 101, 102, etc.
 - d. Check the group id field. It probably won't need to be changed.
 - e. Change the GCOS field. This usually contains the user's full name or any other information you desire. Its use is optional.
 - f. Change the home directory field.
- [2] Check the `/etc/group` file and add the user to the appropriate groups, if necessary. It is usually sufficient to just assign the "users" group ID to the individual in the `/etc/passwd` entry. The `/etc/group` file will only need to be changed, if the user belongs to more than one group.
- [3] Make a new home directory for the user.

```
mkdir /usr/username
```
- [4] Change the ownership of the user's home directory to the new user's name.

```
chown username /usr/username
```

- [5] Change the group ownership of the user's home directory to the new user's group name.

```
chgrp users /usr/username
```

- [6] Change the permissions on the new home directory to 755.

```
chmod 755 /usr/username
```

- [7] Copy a .login and .cshrc file to the new user's home directory. Usually, you can just copy the root's files over or use another user's files.

```
cp /.login /.cshrc /usr/username
```

- [8] Change the ownership of the .cshrc and .login files to that of the new user.

```
chown username /usr/username/.[cl]*  
chgrp users /usr/username/.[cl]*
```

- [9] Use the 'passwd' command to put in an initial password for the user.

The "adduser" script in figure {28}, below, can be used to automatically do all of the above steps. Prior to using it the first time, place this line at the end of your /etc/passwd file:

```
nextuser:xxxxxxxxxxxxxx:101:100:marker for adduser script:/:
```

Change the user id ("101"), if necessary, to reflect the next available user id number. Then, while in super-user mode, execute the adduser script with the new user's logname and full name as arguments, as follows:

```
adduser mike "Mike Flinch"
```

Note that the second argument is quoted because it contains a space.

To remove a user, follow these steps:

- [1] Use "find" to locate all files belonging to the user. Either delete them or change their ownership and move them to an appropriate place.
- [2] Remove the line in the /etc/passwd file for the old user (or change the logname to something like "SPARE1" and put x's in the password field.)
- [3] Delete the old user's home directory and subdirectories.

```

: # force /bin/sh
# Add a new user to the system, must be super-user to use.
# Syntax: adduser logname "fullname"
PASSWD=/etc/passwd
LOGNAM=$1
FULLNAME=$2
HOMEDIR=/usr/$LOGNAM
# Check legality of request and compute params as we go...
if [ $# != 2 ] ; then
    echo "usage:  $0 logname fullname" ; exit 2
fi
if grep "^$1:" $PASSWD >/dev/null 2>&1 ; then
    echo "Error: \"$LOGNAM\" already exists in $PASSWD"
    exit 2
fi
LASTID=`grep "^nextuser:" $PASSWD | cut -f3 -d:`
if [ "$LASTID " = " " ] ; then
    echo "Error: \"nextuser\" line not found in $PASSWD"
    exit 2
fi
NEXTID=`expr $LASTID + 1`
if grep "^.*:.*:$NEXTID:" $PASSWD >/dev/null 2>&1 ; then
    echo "Error: next id ($NEXTID) already in $PASSWD"
    exit 2
fi
if [ -d $HOMEDIR ] ; then
    echo "Error: home directory ($HOME) already exists"
    exit 2
fi
echo updating passwd file...
cp $PASSWD /etc/old.passwd
ed $PASSWD <<-ENDED > /dev/null
/^nextuser:/
i
$LOGNAM:x:$LASTID:100:$FULLNAME:$HOMEDIR:/bin/csh
.
.+1
s/$LASTID/$NEXTID/
w
Q
ENDED
echo "creating new home directory..."
mkdir $HOMEDIR ; chmod 755 $HOMEDIR
cp /.login /.profile /.cshrc $HOMEDIR
cd $HOMEDIR
chown $LOGNAM . .login .profile .cshrc
chgrp users . .login .profile .cshrc
echo "Operation complete. New user: $LOGNAM uid= $LASTID"
echo "Use \"passwd $LOGNAM\" to enter the newuser password."

```

Figure 28. Adduser script

10.2 Managing Processes

Occasionally, a user will start a program which will lock out his terminal or somehow render his CRT inoperative. The super-user must then "kill" the offending program(s). This is done as follows:

- [1] Execute `ps -e` to see what processes are running.
- [2] Find the process attached to the particular terminal (tty) which is causing the problem.
- [3] Issue `kill pid` where "pid" is the process ID found on the ps display. Then, enter `kill -9 pid`, in case the process requires special convincing to stop.

If no particular problem can be found, kill the user's "csh" (or "sh"), and any children of the shell, which you can find. (A "child" process is one which was started by another process, the "parent". Both child and parent can be identified by referencing the PID and PPID columns in a "ps -ef" display.)

10.3 Setting /etc/motd

The message of the day (/etc/motd) is displayed to each user when they log-in. You can use it to make general announcements about system configuration, company news, etc. Try to keep the messages short and remove them when they are no longer valid.

The "login" banner may be changed by editing /etc/gettydefs. The banner usually identifies the system and is displayed prior to each login attempt.

10.4 Monitoring File System Usage

If you run out of space during normal use, the system will stop. If the system will not reboot from Winchester, boot from floppy and delete some files in order to be able to run again from the Winchester. See section {12.9}.

To prevent a problem from occurring in the first place, monitor the system. Use the "df" command to check on the available free space on the Winchester. The "du" command can be used to count the number of disk blocks consumed by files and directories. A complete system V UNIX file system will consume about 30250 disk blocks (512 bytes/block).

Here are some suggestions on files to remove or programs to check to get more space:

- ✦ Portions of the online manual pages (in /usr/man) can be removed. You can delete only those sections to which you do not often refer.

Chances are, you only use the written manual pages, anyway. This could release about two megabytes. Also, the /usr/games directory and libraries can be deleted or cleaned. This could give you a megabyte or more of space, depending on what files you have. Some distributions have a script, "/etc/make.smaller", which can be used to strip the file system to the bare minimum. Execute it with no arguments to display a menu.

- ♣ Read the following sections for suggestions on finding big files.
- ♣ Be sure the accounting programs are off. This will keep the accounting log files from growing. Execute:


```
/usr/lib/acct/turnacct off
```
- ♣ Check your crontab to see if various log files are periodically being cleaned.
- ♣ Ask your users to purge any garbage files. Put a notice in /etc/motd.
- ♣ Look for old user files, starting in the /usr directory. There may be some large files left behind by departed compatriots. The `ls -lR` command may be useful.
- ♣ Check /tmp and /usr/tmp for unclaimed garbage.
- ♣ Add another Winchester, as described in section {12.11}.
- ♣ Use "tar" or "cpio" to copy old files to floppy or tape, then delete the files on your Winchester. Refer to section {11}.

10.5 Garbage Collection

Many programs use /tmp and /usr/tmp for temporary files. Some programs don't clean up after themselves very well; so, you may find some old files in these directories which you can remove. The following lines could be put in your crontab file to automatically clean the /tmp and /usr/tmp directories of unused files:

```
30 4 * * * find /tmp -atime +1 -mtime +1 -exec rm "{}" ";"
32 4 * * * find /usr/tmp -atime +1 -mtime +1 -exec rm "{}" ";"
```

Also, there may be some "core" image files left in various places as the result of program dumps that users have forgotten to remove. You can use this command to find them:

```
find / -name core -print
```

A UNIX "core" file is just a memory dump. The name is a carryover from the old days when computers used core memory.

10.6 Examining Log Files

There are a number of log files which UNIX maintains to record various events. For example, /etc/cron keeps a log file, /usr/adm/cronlog or /usr/lib/cron/log, a piece of which was shown in an earlier section. These files should be examined from time to time to see if the associated programs are functioning properly. The log files should be cleared out occasionally so they don't grow too big - a good function for cron. (See section {6.8.1}.) If you want some ideas on getting more disk space, look in section {10.4}.

Here is a partial list of UNIX log files.

/usr/lib/cron/log	/etc/cron activity log (V.2)
/usr/adm/cronlog	/etc/cron activity log (V.0)
/usr/lib/spell/spellhist	records misspelled words
/usr/spool/uucp/LOGFILE	uucp system activity log
/usr/spool/uucp/ERRLOG	uucp system error log
/usr/spool/uucp/SYSLOG	uucp system system log
/usr/spool/uucp/AUDIT	uucp system log
/usr/spool/lp/log	LP spooler activity log
/usr/adm/sulog	/bin/su log
/usr/adm/acct/...	misc accounting log files
/usr/games/...	misc game logs
/etc/wtmp	login log (non-ASCII)

Figure 29. log files

Since you can use the "find" command to locate any files having particular attributes, you can search for growing log files which you may happen to miss. For example:

```
find / -size +400 -print
```

will inform you of any files larger than 400 blocks (200K bytes). If your file system runs out of space, the system will stop. You will have to reboot (from floppy), mount the Winchester and clean the file system of unnecessary files.

10.7 Setting the Date and TZ

To set the date or time, you must be the super-user and you should be in single-user mode. The command format is:

```
date MMDDHHMM
```

where "MMDDHHMM" is the (numerical) month, day, hour and minute. For example, "date 12051310" would be December 5, 1:10 pm.

- ◆ If you must set the date or time while in multi-user mode, you may want to kill /etc/cron first. This is to prevent cron from going

berserk trying to "catch up" to the new time. After the new time has been set, you can restart cron by entering `~/etc/cron`. If you are only adjusting the time by a few hours, this should not be necessary.

UNIX keeps track of the time in an internal format based on the number of seconds from January 1, 1970. When you do a "date" the time is converted to your local time based on the value of the "TZ" variable in the environment. The TZ variable is set in a number of places, such as:

```
/etc/rc
/etc/bcheckrc
/etc/profile
/etc/cshrc
/usr/lib/uucp/uushell
```

and should have one of the following formats, depending on your own zone:

```
TZ=EST5EDT           TZ=JST-9   (for Japan)
TZ=CST6CDT           TZ=GMT0    (for England)
TZ=MST7MDT           TZ=AST-10  (for Sydney)
TZ=PST8PDT           TZ=GST-1   (for Germany)
TZ=HST10HDT
```

The number is "hours", relative to Greenwich, and may be a positive or negative integer.

If the output of "date" is not correct for your time zone, edit the TZ line in the files listed above, and wherever else you find the TZ variable being set. You should then log out and back in again to check if it was changed properly.

UNIX has U.S. legislative action built in (unfortunately) and assumes everybody observes daylight saving time on the same schedule. If you live in an area which does not observe daylight saving time or if congress changes the law, then set TZ as in this example:

```
TZ=MST7
```

and manually adjust the numeric value when daylight saving starts or stops.

If you want to find the time in another time zone, say Germany, enter the following command (include the parens to protect your own TZ variable from being changed).

```
( setenv TZ xxx-1 ; date )
```

10.8 Checking the Nodename

If you are using the uucp or Ethernet logic, you must be sure your system "nodename" is correct. This is the name by which your system is known to your neighbors on the network. The initial value is compiled into the /unix file. To check it, enter `'uname -n'`. To change it in /unix enter `'/etc/chgnod newname'`. You must reboot for the new name to take effect. You only have to run chgnod once for each new /unix file. If you copy a /unix file from one machine to another, remember to adjust the nodename in the copy.

It is possible to change your active nodename without rebooting by adding these lines to the end of the /etc/chgnod script:

```
echo changing /dev/mem
adb -k -w $kernel /dev/mem << F | sed -n \
    -e '4s/_utsname+0x9/old name/p' \
    -e '$s/_utsname+0x9/new name/p'
utsname+9/s
utsname+8/w $V
utsname+9/s
F
```

Figure 30. /etc/chgnod change

(Note: To add these lines, just copy the existing "adb" command already in the /etc/chgnod script, add the "/dev/mem" argument and change the three adb "?" commands to "/".)

10.9 Running Vchk

The /etc/vchk program will check the directories and files on your system to be sure they have the proper permissions and sizes. Vchk uses a file called "vchk_tree" which contains information about how the system should look. It compares that information with the actual files on the system and prints a list of discrepancies. Vchk will even generate a list of the commands needed to correct any problems. The vchk_tree can be created (by vchk) and examined or modified (with an editor).

The following command will check the system and create a script of repair commands in file "vchk.sh":

```
vchk -Sc > vchk.sh
```

Edit the "vchk.sh" file to remove any commands which are inappropriate for your system (specifically, the UniSoft "take" command or any attempt to read a file from a network). There are many options which you can use with "vchk". Refer to the vchk pages in the UNIX Administrator's Manual.

Do not be alarmed by many of the "errors" which vchk finds. It does a very thorough job of checking your file system and will complain about

files which have grown (like /etc/passwd) or appeared since the system was built (such as log files). If you like, you can recreate or edit /etc/vchk_tree to account for your special circumstances.

10.10 Init

The /etc/init program is the source of all UNIX life. When the system is booted, the kernel loads and starts /etc/init, which, in turn, is responsible for starting all other activities. Init uses /etc/inittab to control which other processes are started. (See section {6.2}.) The normal configuration causes init to enter the "s" state (single-user) until specifically commanded to do otherwise. When the system is in the single-user state, the system administrator can run "fsck" and other programs to check or repair the file system, without interference from other processes.

To switch to multi-user mode, enter `init 2`. Run level "2" is, by convention, the multi-user state. There are other numerical states as well, which are not assigned. The `telinit` command is used to enter states "a", "b" or "c". By adjusting /etc/inittab, you could use state 3, for example, to enable getty's on certain tty ports which are not used in level 2. Study the distribution /etc/inittab and the manual pages for inittab in section four of the UNIX User's Manual to learn more about how to use init.

The command `init s` will send you back to single-user mode. When doing so, init will link the "/dev/syscon" device to the tty port which issued the "init s" command. This is to allow you to continue issuing commands while in the single user mode, since /dev/syscon is the "virtual" system console.

Normally, the system administrator should use the physical "console" device (port B) when using init. If you use a different tty port, then /dev/syscon will be changed, and, if you later reset and reboot, the /dev/syscon device will still be linked to the non-console tty. This may cause some confusion when you reboot, since you will not receive all of the normal boot messages on the console. You can try hitting the "DELETE" key at the console device to get control back. You could also boot from floppy and remake the /dev/syscon device as follows:

```
mknod /dev/syscon c 0 0
```

The /dev/systty device is used by the init program as a reference for the physical system console. That is, when the init program wants to know where the default console should be, it links /dev/syscon back to /dev/systty. Thus, you should have /dev/systty linked to /dev/console.

10.11 Adjusting /etc/inittab and /etc/rc

If it is necessary to modify /etc/inittab (to change serial port configurations) or /etc/rc (to change multiuser mode setup), certain steps are required.

The /etc/init program only reads its control file, /etc/inittab, when the init command is executed. Thus, after /etc/inittab is modified, you must execute at least `init q` (as super-user) to instruct init to reread the modified table. Otherwise, your changes will not take effect. Also, any lingering getty's which you turned "off" will have to be killed manually, via `kill -9 pid`. See section {6.2} for an example inittab.

/Etc/rc is only executed if you change run levels. That is, if you are in run level two (normal multi-user mode), you must go to run level three (or four, etc) to cause /etc/rc to be executed. Also, if you change the level two entry in /etc/rc, you do not want to re-execute that entry unless you are in single-user mode. Thus, the best way to test level two in the modified /etc/rc script is to either `sync`, `sync` and reboot, or return to single user mode via `init s` and kill off all daemons and getty's. Then, enter `init 2`.

It is common to use run levels greater than two to start optional system features. For example, the Ethernet daemons could be started via /etc/rc from init level three. In this case, to restart Ethernet after entering level three, you must first go to some other level (e.g., four) then back to three.

10.12 Mail Aliases (System V.0)

On a system which connects to other machines via uucp or Ethernet, you can control mail delivery via the /usr/lib/aliases file, which allows you to specify routings. For example, the entries:

```
fred:  sm3!fred
sales: jim,sales!tim
adm:   yourname
root:  yourname
```

will route mail for "fred" to "fred" on machine "sm3". Mail for "sales" will be sent to two destinations. This helps local users, since they do not have to specify the mail routing for to people on other machines. The last two lines reroute any mail for "adm" and "root" to go to your mailbox. The /etc/delivermail program is used by /bin/mail to check the aliases file for routings.

Also, a local mail file, such as "/usr/mail/fred" which starts with a line such as:

```
Forward to sys3!fred
```

will cause the mail program to route mail for "fred" on the local system to "fred" on system "sys3". This method, however, requires "fred" to have an account on the local machine.

10.13 System Security

This section outlines some items to watch for when setting up your UNIX system. If you need additional information, look in the UNIX manuals and the reference books listed at the end of this guide.

- ♣ Whenever the system is in single-user mode, the system is very insecure. This is because the default single user is the super-user and no password is required. You should have the system and the "console" device in a secure area to prevent somebody from resetting the system and using the console in single-user mode. You can also modify /etc/inittab so that the system automatically goes to multi-user mode, although this is not recommended.
- ♣ The /etc/passwd file allows passwords to be automatically expired (to force users to change their passwords) or locked (to prevent a password from being changed by anyone but the super-user). See section four of the UNIX User's Manual.
- ♣ All lines of the password file should have an entry in the password field if your system is connected to the phone lines or local network.
- ♣ All files and directories should be checked for proper ownership and modes. You can use the "vchk" program to check most of the system files and directories. See section {10.9}.
- ♣ To secure mail files from being read by others, create a (possibly empty) mail file for each user (and owned by that user) with mode 600. You could also put these commands in /etc/cshrc:

```
touch $MAIL
chmod 600 $MAIL
```

The mode for the /usr/mail directory should be 755.

- ♣ The disk devices, /dev/mem and /dev/kmem should not be accessible by ordinary users. They should be owned by "check" with mode 600.
- ♣ Certain users can be "restricted" by using special home directories and the restricted shell and editor, "rsh" and "red".
- ♣ You can look through /usr/adm/sulog to find who has changed user id's after logging in. However, the log can be doctored by a sneaky user.
- ♣ You can use "find" to search for programs with the "suid" bit set. These programs can masquerade as the super-user.

- ♣ You can use the accounting programs to monitor who is doing what and when. Execute:

```
/bin/su - adm -c /usr/lib/acct/startup
```

or put it in your `/etc/rc` script. Use `'acctcom'` (and others commands) to display the acquired information.

10.14 System Backups

The administrator is usually responsible for backing up the file system on a routine basis. There are two fundamental methods of backing up system files.

- [1] Copy the entire system or important/changed files to media, such as floppy or tape. Details on using media can be found in section {11}.
- [2] Copy the file system (or just selected files) to another partition of the Winchester or to another Winchester. For example, a single `'dd'` command could be used (in single user mode!) to make a "clone" of the root file system on another partition or drive. The standard partition sizes have been selected so that partition "c" can be backed up on partition "d" and "f" on "g". (Refer to sections {12.10} and {12.11} for more details on Winchester drive configuration.) Although this method does not require any media handling, it does consume extra Winchester space.

10.15 Other Things to Watch

- [1] The system has certain limitations on the number of active processes and open files. You can display the current usage and maximum values with the `'pstat'` command. (Reconfiguration Rights, described in section {13.3}, will allow you to change these parameters.)
- [2] Programs that use the shared memory, message and semaphore inter-process communication features of UNIX must release these facilities before exiting. Failure to do so will consume memory and queues which could impair later operation or, in the case of shared memory, cause the system to run out of storage. If you forget, the system will automatically release them if your program exits normally. However, if you are testing a buggy program, things may get left behind. Use the `'ipcs'` and `'ipcrm'` commands to report and control these facilities.

11. USING FLOPPY DISKETTES AND TAPE

11.1 Floppy Disk

CAUTION ITEMS:

- ♣ Do not touch the surface of the media.
- ♣ Do not manually rotate the diskette in its jacket. The jacket is specially designed to remove dust particles from the diskette and manual movement will defeat the cleaning action.
- ♣ Do not store your diskettes in high temperatures.
- ♣ Keep your diskettes away from radiation such as power supply magnetic fields or CRT high voltage transformers. Diskettes can go through airport x-ray equipment; however, the magnetic fields, used to produce the x-rays and operate the conveyor belt motors, may be harmful.
- ♣ Remove your diskette from the drive when not in use.

Note: Refer to section {14.3} for help in interpreting floppy I/O errors.

11.1.1 Formatting a Floppy Diskette

To format a floppy diskette, follow these steps:

- [1] Check that the diskette is not write protected. For a 5-1/4" diskette, the write protect hole must be uncovered. For a 8" diskette, the write protect hole must be covered. (An example of Man's infinite wisdom.)
- [2] Insert the diskette into the drive and close the door. The label should either face to the right or up, depending on the orientation of your drive.
- [3] Use either `fdref(1)` or `diskformat(1)`, depending on the system and the type of diskette you are using. (You must be super-user on some systems.)

<u>Command (for SBX-FDIO only)</u>	<u>Size</u>	<u>Sides</u>
<code>fdref /dev/rf5sd -d 4</code>	5-1/4"	single
<code>fdref /dev/rf5dd -d 4</code>	5-1/4"	double
<code>fdref /dev/rf8sd -d 7</code>	8"	single
<code>fdref /dev/rf8dd -d 7</code>	8"	double
<code>fdref /dev/rf5dh -d 7</code>	5-1/4" HD	double

<u>Command (for OMTI-5400 Floppy)</u>	<u>Size</u>	<u>Sides</u>
diskformat /dev/rf5sd -il 4 -head 0	5-1/4"	single
diskformat /dev/rf5dd -il 4	5-1/4"	double
diskformat /dev/rf8sd -il 7 -head 0	8"	single
diskformat /dev/rf8dd -il 7	8"	double

Note that the "raw" floppy device is used. See the fdref(1) manual page at the end of this guide or diskformat(1M) for details.

11.1.2 Writing Data to a Floppy Disk

- [1] Check that the diskette is not write protected. For a 5-1/4" diskette, the write protect hole must be uncovered. For an 8" diskette, the write protect hole must be covered.
- [2] Insert the diskette into the drive and close the door. The label should either face to the right or up, depending on the orientation of your drive.
- [3] If the diskette has never been used before in the system, it will probably need to be formatted. See the previous section for details.
- [4] Enter one of the following commands:

<u>Command</u>	<u>Size</u>	<u>Sides</u>
tar -cvlf /dev/rf5sd files	5-1/4"	single
tar -cvlf /dev/rf5dd files	5-1/4"	double
tar -cvlf /dev/rf8sd files	8"	single
tar -cvlf /dev/rf8dd files	8"	double

In the above table, "files" is a file name, a list of file names or a directory name (including ".").

If you specify the size of the diskette (see section {11.1.5}) and if no single file is larger than one diskette, then "tar" will allow a large directory to be split between diskettes. It will prompt you when to change disks. Refer to Section one of the UNIX User's Manual for information on "tar".

The "cpio" program could have been used as well, as in:

```
cpio -ocv files > /dev/rf5dd
```

There are more examples of "cpio" backup techniques in the sections on the streamer tape, below.

11.1.3 Recovering Data from a Floppy Disk

- [1] Insert the diskette and enter one of the following commands:

<u>Command</u>	<u>Size</u>	<u>Sides</u>
tar -xvlf /dev/rf5sd files	5-1/4"	single
tar -xvlf /dev/rf5dd files	5-1/4"	double
tar -xvlf /dev/rf8sd files	8"	single
tar -xvlf /dev/rf8dd files	8"	double

This will extract the named files (or directory) and put them in your current directory, unless the data on the diskette specifies a full pathname starting at "/", the root.

⚠ Caution: be sure you are in the desired directory before issuing this command. Otherwise, you may overwrite important files.

- [2] You can list the contents of a diskette without actually reading the data by using a "tar" command of the following form:

```
tar -tvlf /dev/rf5dd
```

11.1.4 Other Ways of Using a Floppy

There are other methods which may be used to put data on a floppy diskette. We do not recommend that you use these techniques unless you have special circumstances.

- a. Use the "dd" command to write or read data. This treats the diskette as one giant file. You can skip or seek to certain blocks on the diskettes; but, it is slow going. This command is generally only used to look at the data on a diskette for debugging problems or for writing the standalone boot program on the first few blocks. Example command:

```
dd if=/dev/rf8dd of=/tmp/file count=3
```

- b. You can use the `mkfs1b` command to make a file system on the diskette (which has a directory structure, etc.) and then `mount` the diskette and treat the files on it as part of the UNIX file system. This structure allows random access. For sequential file access, use `tar` or `cpio`. Example mkfs commands are:

```
mkfs1b /dev/f5dd 1440      (for 5")
mkfs1b /dev/f8dd 2310     (for 8" DD)
```

The "mkfs1b" uses 512-byte logical sectors, whereas "mkfs" uses 1024. Also, note that the block device name was used.

- c. If you're really lazy or interested in doing a quick test, you can "cat" directly to and from the device, as follows:

```
cat file > /dev/rf5dd
cat /dev/rf5dd > newfile
```

When the file is read back from the device, it will have garbage at the end because the end-of-file will be at the end of the device. In this case, the whole floppy will be read back in, unless you hit the "DELETE" key.

11.1.5 Diskette Capacity

The following chart shows the capacity of a floppy diskette, in blocks. One "block" contains 512 bytes. The UNIX kernel works with 1024 bytes at a time, but maintains block numbers in multiples of 512 bytes. The double density (DD) format block size may be specified by the fdref command. The following table assumes 512 byte physical blocks for DD and HD (high density) and 128 byte physical blocks for single density (SD) diskettes. All formats are soft sectored.

<u>Type</u>	<u>dens</u>	<u>sectors/track</u>	<u>trks/side</u>	<u>sides</u>	<u>BLOCKS</u>	<u>device</u>
5"	DD	9	80	2	1440	/dev/rf5dd
5"	HD	15	80	2	2400	/dev/rf5dh (M10)
8"	DD	15	77	2	2310	/dev/rf8dd
8"	DD	15	77	1	1155	/dev/rf8sd
8"	SD	26	77	1	500	/dev/rf8ss

Table 6. Floppy Diskette Capacities (Blocks)

The HD format (for IBM PC diskettes) requires the use of a special SBX-FDIO module. Support for this format is available only on the HK68/M10.

If you are using the "tar" -B option (which specifies media size), this is the proper format:

```
tar -cvlfB /dev/rf5dd 1440
```

Other formats may be used. Refer to the fdref(1) manual page (at the end of this guide) or the standard diskformat(1) page for details.

11.1.6 Floppy Media

Do not use "quad density" or "high density" diskettes for a double density recording, even if they say "96 tpi" on the label. Contrary to what your local computer dealer may tell you, they are not compatible with double density diskettes. "High density" diskettes, however, must be used with the high density format. If you use the wrong type of diskette media, you will get frequent read and write errors.

11.1.7 Raw Devices

Although the block device (such as "dev/f5dd") could be used for all floppy operations except formatting, the raw device (such as "/dev/rf5dd") is usually a better choice. I/O errors, such as no media or physical end-of-file are reported more promptly by the raw device. This is important if you need to store a large file across multiple diskettes since the physical end-of-file needs to be reported to tar or cpio. The block device must be used only with "mkfs" and "mount".

11.2 Streamer Tape

A streamer tape may be used to efficiently save and restore large amounts of data. The data transfer rate is high and the tapes have a large capacity (about 40 megabytes). Tape capacity (in megabytes) is approximately equal to the tape length (in feet) divided by 10.

There are two logical streamer tape devices assigned to the drive.

- ✦ Device `"/dev/st1"` will rewind the tape after each use.
- ✦ Device `"/dev/st0"` will seek forward to the next file mark after each command. `"st0"` allows additional files to be appended to a tape.

There are also a few utility commands which control the streamer drive:

<code>strewind</code>	Rewind the tape
<code>stretension</code>	Run forward and back
<code>sterase</code>	Erase tape
<code>ststats</code>	Display drive status

You will find manual pages for these commands and more details on the streamer interface at the back of this guide (see ST(7).)

Although the following procedures utilize the `"cpio"` program, `"tar"` could be used as well with only slight variation in the command format.

Refer to section {14.4} for help in interpreting streamer tape I/O errors. Common errors are due to the cassette not being in place, the write protect setting being on `"safe"` or the tape having wound off the cassette reel.

CAUTION ITEMS:

- ✦ Do not touch the surface of the tape as oil and grease can damage the tape and the drive heads.
- ✦ Do not manually crank the tape in the cassette. If it is not positioned properly, it could be wound off the takeup reels by the controller.
- ✦ Do not remove the cassette while the tape is in motion. If the LED on the drive is off, then it is okay to open the door. If the LED is on, then the tape may or may not be in motion, depending on the command you issued. Also, the tape may be in motion even though you have a prompt. If you are unsure, listen closely to the drive for activity before you open the door.

11.2.1 Writing Data to Streamer Tape

- [1] Make sure the tape cartridge is not write-protected. The write protect adjustment in the corner of the cassette must not be on "safe". Insert it into the drive and close the door.
- [2] Erase the tape by entering `sterase`. Or, if the tape is already erased, enter `stretension` to run the tape to the end and back. This step insures that the tape is correctly tensioned in the cartridge, which will give you a more reliable recording.
- [3] Write to the tape by using a command sequence of the following form:

```
cd wherever
find . [-other args] -print | cpio -ocvF > /dev/st1
```

The "find" command will generate a list containing the name of every file in the system which meets whatever constraints you place in the "find" command. Each name is passed to the "cpio" command. The options on cpio are "o" for output, "c" ascii headers (to improve portability between UNIX implementations), "v" for verbose, and "F" which forces a 32K byte buffer. You should do this when the system is relatively calm, since any files which are changed while the tape is being written may not be copied correctly.

The use of `cd` and the "." pathname in `find` causes find to generate file names which are not anchored at any particular place. This allows more flexibility, since files which are later retrieved from tape can be placed wherever desired.

The "cpio" program will respond with a list of the file names that are transferred to tape and the total number of blocks written.

- [4] The tape will automatically rewind when the copy is completed, since /dev/st1 was used.

A `tar` command may also be used with the streamer, in which case the syntax for creating a tape would be:

```
cd wherever
tar -cvlfb /dev/st1 64 files
```

The `tar -r` and `-u` options are not supported when using a streamer tape.

Here is an example of a script which will backup your files. When run without an argument, `backup` will write to tape any files which have been modified since the last backup was done. You can, instead, provide the number of days to look back for updated files. You should be super-user when you run this command to be sure you have access to the tape device and all directories and files in the search path.

```

: # backup files to streamer # "backup [days]"
stoplist="^\./dev|^\./usr/spool/|^\./tmp/"
base=/
daily=/usr/adm/last_backup
daily_new=/usr/tmp/backup
rpt=/tmp/backup.rpt
if [ $# != 1 -a $# != 0 ] ; then
    echo usage: $0 [days]
    exit 1
fi
if [ $# = 0 ] ; then
    echo Doing daily backup...
    if [ ! -f $daily ] ; then
        echo No basis file found, creating...
        touch $daily_new
        echo Backing up over past 2 days.
        cmd="-mtime -2"
    else
        echo "Backing up since: "\
            "`ls -l $daily | cut -c42-54`"
        touch $daily_new
        cmd="-newer $daily"
    fi
fi
else
    echo Backing up over past $1 days.
    touch $daily_new
    cmd="-mtime -$1"
fi
echo Insert tape and hit CR \\c
read response
echo Stoplist = $stoplist
sterase
cd $base
find . $cmd -print |\
    egrep -v $stoplist > $tmp1 |\
    cpio -ovcF > /dev/st1 2> $rpt
lp -c -t"Backup Report" $rpt
rm -f $tmp1 $rpt
echo Backup complete.
echo "Okay to update last_backup_time (y/n)?" \\c
read response
if [ "$response" = y -o "$response" = "" ] ; then
    mv $daily_new $daily
fi
echo Remove tape when rewound.
exit 0

```

Figure 31. Script for Incremental Backup

The "stoplist" function could be provided by a file instead of a shell variable, if desired.

11.2.2 Recovering Data from Streamer Tape

- [1] Insert the tape and close the drive door.
- [2] Do a "cd" command to position yourself in the directory where you want to put the files.
- [3] You can list the contents of a tape via:

```
cpio -itmvcF < /dev/st1
```

- [4] To read a tape, enter:

```
cpio -icmvdF files < /dev/st1
```

Cpio will read the specified "files" from the tape and place them into your current directory. If no files are specified in the command, all files on the tape will be read.

⚠ Caution: be sure you are in the desired directory when you read from the tape. Otherwise, you may overwrite important files.

11.2.3 Other Streamer Notes

If the streamer is connected to an OMTI 5400 controller and you wish to use a UNIX utility other than cpio or tar, you will have to provide a data buffer in your utility or use the dd command, as in:

```
cat files | dd of=/dev/st1 obs=10b
```

Otherwise, the drive will not "stream".

11.3 Reel-to-Reel Tape

The method used to store and retrieve data from a reel-to-reel tape is similar to the streamer tape procedures detailed in the previous section. Either "tar" or "cpio" may be used. Refer to section {14.2} for a description of tape I/O errors.

The special notes which apply to reel tape operations are:

- ♣ A tape may be write protected by removing the plastic "write ring" from the reel hub. The drive senses if the ring is in place when you mount the tape.
- ♣ The following commands may be used to position the tape. You will find manual pages for these at the back of this guide.

```
mtrewind    rewind tape
mtskip      skip file marks
```

- ♣ The "tar" command, if used, must always have the blocksize specified. That is, the tar "b" option flag (and blocksize argument) must always be used. For example, this command will write files to tape:

```
tar -cvfb /dev/mt0 20 files
```

"Files" can be a name list or a directory name (including ".").

- ♣ The "cpio" commands for reel-to-reel tape are similar to the commands used for a streamer tape. The only difference is the device names, and you may need to use the "B" option (for a 5K buffer) instead of a "F" 32K buffer). Refer to the previous sections on using a streamer.
- ♣ Large dumps can be split between tapes. With cpio, this is done automatically done. With tar, you will need to specify the "s" or "B" options; see tar(1).
- ♣ Typical 1600 BPI tape devices are:

```
/dev/mt1    rewinds the tape after each command.
/dev/mt0    does not rewind the tape after each command.
```

For complete details on the tape devices, refer to the mt(7) manual page at the back of this guide.

Approximate tape capacities (megabytes) are as follows:

<u>BPI</u>	<u>2400 ft</u>	<u>1200 ft</u>	<u>600 ft</u>
800	20 meg	10 meg	5 meg
1600	40 meg	20 meg	10 meg
3200	80 meg	40 meg	20 meg

11.4 Media Interchange

If you are exchanging media between a Heurikon system and another UNIX system, keep these items in mind:

- ⊕ Standard floppy format is 128 bytes/sector, 26 sectors/track (single density) or 512 bytes/sector, 15 sectors/track (double density).
- ⊕ Byte swapping: There is, unfortunately, more than one standard byte orientation for tapes. The "normal" convention is to write the high byte of a 16-bit binary value followed by the low byte. The other convention (DEC) is to write the low byte first. This would not be a problem if it weren't for the fact that in both cases character strings are always written in normal byte order. Thus, when a tape is read, you must know whether you are reading binary or character data in order to determine which bytes should be swapped. You can deal with this problem the following ways:
 - a. Read the data to a buffer and write a program to swap whichever bytes need to be swapped. This will work if you know in advance the exact byte structure of the tape data.
 - b. Use the `'conv=swap'` option on the `'dd'` command to swap all byte pairs.
 - c. Specify the swapping minor device names for the (reel-to-reel) tape. See the `mt(7)` manual page at the back of this guide.
- ⊕ Most systems support a similar "tar" format. Use "tar" instead of "cpio" to create tapes for other systems.
- ⊕ Use the same data blocking factor on both machines. That is, the tar `-b` option or the cpio `-B` option must correspond. Values larger than 20 with tar or use of the `-F` option with cpio will probably not work. The `mtforeign(1)` command may help you determine the format of a tape; see the manual pages at the end of this Guide.
- ⊕ After reading in foreign files, check their modes (e.g., user id), to be sure they are correct for your system.

Another option is to use Uucp to transfer files between systems.

Programs which were compiled on a Heurikon System III machine should be recompiled in order to run on System V or V.2. Streamer tapes written on a Tanberg drive may not be readable on a Kennedy drive.

11.5 Backups via Ethernet

Files can be sent between machines very efficiently using Ethernet links. The following command examples allow files to be saved on remote tape facilities:

```
tar cfb - 20 files | remsh sys2 dd of=/dev/mt0 obs=20
find . -print | cpio -ocvB | remsh sys2 dd of=/dev/mt0 obs=10b
```

11.6 Method Comparison

There are various methods described above for using media to backup UNIX files. Here is a summary of those methods and their relative advantages.

file system	Use the mkfs1b command to create a file system on the media. Allows random access to files. Places a directory structure above the raw data. Cannot be used with streamer or reel-to-reel tape. Best for a situation where multiple files will be placed on a floppy and a section of a file must be updated at a later time. Data will not be contiguous on the media.
tar command	Sequential access method. Can handle a whole directory and sub directories by specifying only one directory name. Cannot handle special files (devices). The most standard inter-system format.
cpio command	Sequential access method. Requires a list of file names to be fed to the command. Can handle special files (devices). Will automatically request a volume change at the end of the media.
dd command	Sequential access method. Allows access to specific blocks on the media. Does not interpret any data or place any meaning on specific records.
cat, see	Since a UNIX device is treated like an ordinary file from the point of view of a user program, any utility (e.g., cat) can do I/O directly. Not very elegant, but functional.

12. REBUILDING THE UNIX SYSTEM

Rebuilding a UNIX system is defined as:

- ♣ Reformatting (reinitializing) your Winchester disks.
- ♣ Reexamining your Winchester disks for defects.
- ♣ Recreating an empty UNIX file systems on your disks.
- ♣ Reloading these new file systems with files from a previously existing file system.

Rebuilding the UNIX system can take three or four hours from floppy disk or half an hour by streamer tape. Do not rebuild the system unless it is absolutely necessary, such as, after replacing a Winchester drive or after experiencing a major hardware problem, which resulted in many corrupted files.

If you cannot boot, then you may only have a bad standalone boot program. That can be rewritten, as described in a section {12.4}, without rebuilding. A few lost files can be retrieved by specifying them in a "tar" command instead of reading in all of the UNIX files. If you have new bad blocks on the Winchester, the 'badblk' program can be used to assign alternates. Refer to section {12.5}. Rebuilding should be considered only as a last resort.

- ♣ ASK YOURSELF "IS REBUILDING REALLY WHAT YOU SHOULD BE DOING?" If you are unsure of the answer to this question, call Heurikon Technical Support for advice rather than make the wrong choice.

When you rebuild the system, you will overwrite all the files on your system and lose any recent work. Of course, if you have backup diskettes or tapes containing your recent file changes, then you will be able to load those back in after rebuilding.

- ♣ CAUTION: If the system is off, power it up and wait at least an hour for it to come to a stable temperature before rebuilding. This is important because the Winchester drive should not be formatted until it has had a chance to stabilize; otherwise, you may have trouble reading from the drive later.

The diagram on the next page shows the general relationships between the various diskettes and the Winchester drive with respect to the rebuild procedure.

12.1 Floppy or Tape Rebuilding

Use this sequence to rebuild the UNIX system from floppy disk, streamer or reel-to-reel tape.

- [1] You will need the following items to rebuild the system: (Refer to figure {32}.)
 - a. The "boot" diskette. This diskette has the standalone boot on the beginning and nothing else. It is a separate diskette so that the "rebuild" diskette has more space for files.
 - b. The "rebuild" diskette, containing the minimum set of programs and files needed to run UNIX, format the Winchester, create a file system, copy files and debug problems. In particular, it has a floppy-based /unix for execution, which is modified and moved to the Winchester to become a Winchester-based /unix, a standalone boot for copying to the Winchester, /etc/init, /etc/inittab and numerous programs in /bin. See section {12.9} for more information.
 - c. One "cpio" format tape or streamer tape cassette or a set of from 15 to 25 diskettes containing all of the standard UNIX files.
 - d. Any special language diskettes and your own file backup diskettes or tapes.
- [2] Push the system reset button. See the "CAUTION" note above.
- [3] Insert the "boot" diskette in the drive.
- [4] Enter `^bf^`. You should get the "standalone boot" message from the loader.
- [5] Remove the boot diskette and insert the "rebuild" diskette.
- [6] Enter `<CR>` to load the floppy-based /unix. This will take a few seconds.
 - ♣ With older rebuild scripts, you may also get a "Type RETURN to start at 0x1000" message. If so, do so.
- [7] Enter `<CR>`, again. This will start execution of the floppy /unix.
- [8] If you are going to rebuild from tape, then insert the "cpio" streamer cassette in the tape drive and close the door or mount the reel-to-reel tape.
- [9] Enter `^/rebuild^`. This will run the rebuild script on the floppy diskette. You can print this file (via `^cat^` or `^dd^`) for specific

details. You will receive a menu with some options to choose.

♣ Note: Some system releases have a rebuild procedure which varies somewhat from that described below. Follow the specific instructions provided with your rebuild media or as prompted by the rebuild script. Use this document as a guide to what will be happening.

- [10] You will be asked if you want to do a "dry run" instead of the real thing. On your first time through, say 'yes'. This will cause all format and rebuild commands to be displayed but they will not be executed. Later, you can repeat this procedure to actually perform the rebuild.
- [11] If you are asked "Enter Device (wd/smd)", answer "wd" for a Winchester or "smd" for a SMD based system.
- [12] You will be asked to select the type of controller used in your system (e.g., Adaptec or OMTI). Respond according to the menu choices.
- [13] For each of two drives, you will be asked to select the code for the type of Winchester (or SMD) in your system and the partition you wish to build. Enter the code for your Winchester according to the menu choices. There will be a few seconds of disk and CPU activity, after which the computed drive parameters will be displayed.
- [14] You will be given the option of formatting the drive. This activity can be skipped if you are sure the drive is properly formatted and there are no bad blocks. You must skip this step if you want to save data or the file system on any partition of the drive. If you format the drive, you may be able to hear it step from cylinder to cylinder as it goes. Formatting will take a few minutes to complete. When the format operation is completed, a few status messages will be printed.
- [15] After the format operation, the drive will (optionally) be checked for bad blocks and, if any are found, alternate blocks will be assigned. Error messages will be generated as bad blocks are encountered. This step will take about 10 minutes, depending on your drive size.
 - ♣ You must let the "badblk" program run to completion in order for the bad block information on the drive to be updated and alternate blocks to be assigned.
- [16] After the bad block check is done, the rebuild script will write a standalone boot program on the beginning of the drive (drive 0 only) and a UNIX file system will be constructed.

- ♣ (For releases prior to 7a only): You will be asked whether you want to rebuild from floppy or tape. Respond `^f^` or `^t^`. If you only want to place an empty file system on the partition, answer `^t^`.
- [17] A "mini" version of UNIX, consisting only of the kernel and a few vital commands, will be placed on the Winchester.
- [18] When you get the message "Finished rebuilding" (or "done") and a UNIX prompt, enter two `^sync^` commands. This will make sure the Winchester superblock is updated.
- [19] Push the system reset button. This will return you to the Hbug monitor.
- [20] Enter `^bw^` to boot from the Winchester (or `^bd^` for SMD). You will get a "Standalone boot" message. (Note: from here on, if you have trouble, refer to the boot procedure described in section {1.1} for troubleshooting hints.
- [21] Enter a carriage return. The boot program will read the kernel from the Winchester. This will take up to a minute.
- ♣ Enter another carriage return if you get "Type RETURN to start..."
- [22] After a minute or two, you will get the standard messages from UNIX as the system is initialized. When you get the UNIX prompt ("`#`"), the system will be ready to load the rest of the UNIX programs.
- [23] Enter `^loadsys^`. This script will load in the "tar" format diskettes or "cpio" format streamer tape containing all of the UNIX programs and files.
- ♣ Prior to release 7a, `^loadsys^` was called `^loadtar^`.
- [24] When asked to do so, insert the the streamer tape (or diskette volume 1) in the drive and enter a carriage return. The name of each file will be listed as it is read from the media and loaded onto the Winchester.
- ♣ If you are not building the root file system or if no files need be put on the partition, hit the "DELETE" key and go to step the next step (or back to step 9 to do another partition.)
 - ♣ Loading from streamer tape will take about half an hour. When the tape has been read in, you will receive the standard UNIX prompt.
 - ♣ If you are rebuilding from reel-to-reel tape, when the message instructing you to insert a streamer tape appears, hit the

"DELETE" key, which will stop the loadsys script. Then, physically mount your tape and enter the following commands:

```
cd /
cpio -icvBdu < /dev/mt1
rm -fr /lost+found
mklost+found
```

- ♣ If you are loading from floppy, the system will prompt you to change diskettes as each volume is completed. Insert the next diskette and type a carriage return. When all the volumes have been read in, respond `q`.

If you skip any volumes or do not complete the rebuild procedure, you may be missing some important files. Some programs will give strange results if they cannot find all of the library and data files, etc., that they need to operate. You can remove files you do not want after the rebuild has been completed.

- [25] The loadsys script will make the "lost+found" directory, which is needed by `fsck` to correct file system problems.
- [26] If you are using a HK68/M10 system, run `Install`.
- [27] Enter two sync commands.
- [28] Press the system reset button.
- [29] Reboot from the Winchester (see section {1.1}). You will be running the "real" UNIX, placed on the Winchester by loadsys.
- [30] Run `fsck`.
- [31] (Optional) Run the "vchk" program to check for missing files and incorrect file modes. Refer to section {10.9} for details. You can do this step later, if you like.
- [32] Load any language or backup tapes or diskettes using the "tar" or "cpio" commands, as follows:

```
tar -xvfl /dev/rf5dd (for 5-1/4" diskettes)
tar -xvfl /dev/rf8dd (for 8" diskettes)
cpio -icvud files </dev/st1 (for tape)
```

Refer to section {11} for details on floppy and tape backup/restore procedures.

That completes the rebuild procedure. You should be able to boot and run UNIX according to the instructions at the beginning of this guide. Be sure to run the "fsck" portion of the boot procedure.

12.2 Creating Boot and Rebuild Diskettes

A "boot" diskette contains the standalone boot program on the beginning of the diskette. A "rebuild" diskette has a minimum UNIX system on it; just enough to bring in programs from other media.

[1] Enter `^cd /usr/src/rebuild^` to ensure that you are in the proper directory.

[2] Boot Diskettes are made as follows:

```
For 5-1/4" diskettes:  ^make boot5^
For 8" diskettes:     ^make boot8^
Manually:             ^dd if=sbootx of=/dev/f5sd ibs=32 skip=1^
Manually:             ^dd if=sbootx of=/dev/f8sd ibs=32 skip=1^
```

[3] Rebuild Diskettes are made as follows:

```
For 5-1/4" diskettes:  ^make sys5^
For 8" diskettes:     ^make sys8^
```

Those commands use the "make" program to execute a series of commands which will ask you some questions and prompt you to do certain manual things, like inserting a floppy diskette. (Look at "/usr/src/rebuild/Makefile" to see the actual commands that are executed to make boot and rebuild diskettes. You can enter them manually, if you would rather.)

There is also a "fix" diskette, which can be made by typing `^make fix5^` (or `^make fix8^`). This diskette will contain some programs which might be useful when repairing a damaged file system.

12.3 Creating Streamer Tape Dumps

Note: This is not the incremental dump procedure. This section describes how to make a complete system dump, which should be done infrequently. Instructions for doing incremental dumps are in section {11.2.1}.

- [1] Ensure that all mountable file systems are unmounted. Do a `mount` command. Any file systems other than "w0b" must be unmounted. Unmount them using the "umount" command, as in:

```
cd /
umount /dev/wla
```

The reason that only the root file system should be mounted is that the tape back-up will contain every file on the system which is reachable at the time the back-up is made. If the back-up contains files from many file systems in addition to the root file system, it is very likely that when the tape is loaded to reinitialize the root file system, the root file system will overflow.

- [2] Be sure the system is calm. Manually log off all users or enter `init s` at the console.
- [3] Make sure the tape cartridge is not write-protected. Insert it into the drive and close the door.
- [4] Erase the tape by entering `sterase`, or, if the tape is already erased, enter `stretension` to run the tape to the end and back. This step will insure that the tape is correctly tensioned in the cartridge to insure a reliable recording.
- [5] Write to the tape by typing:

```
cd /
find . -print | cpio -ocvF > /dev/st1
```

The "find" command will generate a list containing the name of every file and directory in the system. Each name is passed to the "cpio" command. The options on cpio are "o" for output, "c" ascii headers (to improve portability between UNIX implementations) and "v" for verbose.

The "cpio" program will respond with a list of the file names that are transferred to tape and the total number of blocks that are written.

- [6] The tape will automatically rewind when the copy is completed.

12.4 Writing the Standalone Boot to Winchester

- [1] Boot from floppy, according to section {1.2}.
- [2] The standalone boot can be placed on the Winchester, using one of these "dd" commands:

```
dd if=/sbootx of=/dev/rw0h ibs=32 skip=1 seek=1      (Sys V.0)
dd if=/bootx.bin of=/dev/rw0h ibs=32 skip=1 seek=1  (Sys V.2)
```

The `^sbootx^` file is a copy of either `^ml0boot^` or `^vl0boot^`, which are the unified bootstrap routines for system V.0.

12.5 Bad Block Checking

On a large file system, you should expect an occasional defective disk block. UNIX System V.0 (V.2) deals with this problem by reserving a group of extra blocks (tracks) at the end of the drive, for use as "alternates", in case a bad block is found. The `^badblk^` program can be used to check the drive for bad blocks, assign an alternate for a specific disk block (system V.0) or entire track (system V.2) and list the current bad blocks on a drive.

The bad block program is automatically run by the `^rebuild^` script during the system rebuild process. It takes about 10 minutes to run using the read only option; longer, if you use the write/verify option. System V.0 has a 50 block limit on the number of bad blocks which can be remapped. System V.2 can remap 100 tracks.

12.6 Manual Disk Format/Mkfs

If you want to manually format and create a file system on a Winchester (instead of using the rebuild script), you will have to specify numerous drive parameter values. These numbers are automatically computed for you when you use the `/usr/src/rebuild/rebuild` script, which you should examine for details. You can run the rebuild script in the "dry run" mode to see the values for a particular type of drive configuration. Here are examples of the format and mkfs commands used for a Maxtor 65 megabyte drive, with an OMTI 5400 SCSI controller.

```
diskformat /dev/rwlh -cyl 0-917 -head 0-6 -il 1
chstep /dev/rwlh 0 0      (System V.0 only)
badblk /dev/rwlh
mkfs /dev/w1a 3983 1 119
mkfs /dev/w1b 50571:12642 1 119
```

The physical block size on the Winchester is 512 bytes/sector. The logical block size is 1024 bytes/sector.

12.7 Changing the Swap Space Size

The UNIX operating system uses one partition of one drive as a "swap area". This space is used primarily for temporary storage of a program and its data when some other program needs more RAM space. There is no way to predict the maximum amount of space required, since that value depends on system loading and program sizes. The standard system has about four megabytes (7966 disk blocks) reserved for the swap area. If you have a large system memory (more than two megabytes) this might be too small. If the swap space is too small, you will get some warning messages or your system may "thrash" (drastically reducing performance). The kernel will panic if it runs out of swap space at a critical time.

The following procedure may be used to enlarge the swap space to eight megabytes. Read though all steps before you start.

[1] Backup your files; this procedure will wipe your drives clean.

[2] Make a new rebuild disk. (See section {12.2}.)

```
cd /usr/src/rebuild
make sys5
```

[3] Modify the (new) disk to have a floppy based unix which knows of the new partition boundaries.

```
mount /dev/f5dd /floppy
adb -k -w /floppy/unix
nswap,l?U ; Displays old length of 7966
?W Od15966 ; New length of 15966 blocks
$q ; exits adb
```

[4] Change the rebuild script on the floppy so that it will build the partitions using the larger swap area. (This is not required for release 7a and later; the rebuild script has been taught to adjust itself).

```
vi rebuild
/7966
cw15966<ESC>
ZZ
```

[5] Enter two `sync` commands and reset the system. Reboot from floppy (per section {1.2}) using the new rebuild diskette.

[6] Do a rebuild using the new rebuild disk and the new script. If you have two Winchester drives, you must rebuild (mkfs) the second drive also because the kernel assumes the swap partition is the same size on both drives. The partition boundaries will be adjusted to conform to the layout in section {12.10}.

♣ Some (old) rebuild scripts might imply that partitions "b" and "c" still start at block 8000. Don't worry; the kernel knows.

- [7] Modify the UNIX kernel that you plan on booting from the Winchester (e.g., /unix) as shown below. When the rebuild script exits, /dev/w0b should still be mounted at /floppy.

```
adb -k -w /floppy/unix ; default boot kernel
nswap,l?U ; Displays old length of 7966
?W 0d15966 ; New length of 15966 blocks
$q ; exits adb
```

If the "?U" command displays "15966", that kernel has already been changed. This would be expected if you rebuild from floppy since the Winchester /unix written by the rebuild process is actually a (slightly modified) copy of the floppy /unix.

- [8] Enter two `sync` commands and reset the system.
- [9] Enter `bw` to boot from Hbug. The bootstrap will automatically try look at both blocks 8000 and 16000 to find the root file system.

♣ Older bootstraps require you to enter

```
wd(0,16000)/unix
```

at the ":" prompt enter. If so, from now on, any kernel that you boot will require the "wd(0,16000)". This boot-time argument specifies the beginning of the root file system, which is always at 34 blocks more than the size of the swap area.

- [10] Run the adb command as per the above on all other UNIX kernels on your system.
- [11] If you are using an old rebuild script (prior to release 7a), run the vi command as per the above on /usr/src/rebuild/rebuild.

To adjust the swap space to any particular size, follow the above steps but substitute new arguments where appropriate. Replace the "15966" values with the swap space size in decimal and replace the "16000" value with the swap space size in decimal plus 34.

12.8 File System Check, fsck

When the system is booted, it is always a good idea to run the `fsck` program. This is done automatically when going to multi-user mode. (See section {1.1}). If you run it manually, fsck should be used on a mounted file system (such as the root file system) only when in single-user mode. This is to prevent the file system from being changed by cron or another user while fsck is running. If fsck corrects an error on the root (mounted) file system, you will have to reboot to make the correction

permanent.

The "-s" option of fsck can be used to restructure the free list. This will improve system performance, but only for a few hours, since the random nature of the file system will ultimately return.

12.9 Creative Use of the Rebuild Diskette

In addition to providing the means to rebuild the system in the event of a major crash, the rebuild diskette can also be used to examine, and possibly repair, a damaged file system.

There are some problems, however.

- ♣ Due to space limitations, the diskette does not contain many programs.
- ♣ If the Winchester file system is damaged, chances are you won't be able to do much anyway, unless you're a UNIX wizard, in which case you probably wouldn't be reading this section.

Here are some hints for expanding the usefulness of the rebuild diskette.

- [1] Boot the system from floppy as described in section {1.2}.
- [2] Mount the Winchester at "/floppy":

```
mount /dev/w0b /floppy
```
- [3] Commands which are "not found" on the floppy, such as "ls", "rm" and "cat" can be executed from the Winchester by entering `~/floppy/bin/rm`, etc.
- [4] Some programs on the floppy can be used for other purposes. For example, "dd" can be used instead of "ls", "cat" or "cp"; ">" can be used to clear a file instead of "rm".

The makefile in the rebuild directory can make a `fix` diskette containing programs which are useful for debugging and repairing a file system. To be of any value, however, you must make the `fix` diskette in anticipation of its need.

12.10 Winchester Partitioning

12.10.1 Winchester Partitioning - General

The main Winchester drive is logically separated into four areas. The first is a small area reserved at the beginning of the drive for the standalone boot program. The next section is the "swap area" and is used by UNIX to construct memory images and as a place to put programs which are waiting to execute but which cannot due to lack of memory, I/O in progress or low priority. The third area is the actual file system and is known as the "root" file system. It extends almost to the end of the drive. There is another small area reserved at the end of the drive for the "alternate" blocks. In System V.0, this is 100 blocks, although only 50 are ever used; in System V.2, 1% of the drive is reserved for alternate tracks; 100 of which are used, at most. Any defective (or "bad") blocks in the swap or root file system areas are "mapped" into a good block (or track) in the alternate block (track) area of the drive.

<u>Partition</u>	<u>From</u>	<u>To</u>	<u>Function</u>
	<u>(blocks)</u>		
-	0	0	Drive parameter block
-	1	33	Reserved for boot
a	34	7999	Swap area (standard)
b	8000	alt	Root file system
-	alt	end	Alternate blocks/tracks

The parameter block starts with a branch instruction to the second block. This causes the MPU to jump over the parameter block to get to the bootstrap.

A physical "block" is 512 bytes. The device name "w0b" refers to the "b" partition of drive 0. See section {17.3.3} and {17.4} for details on minor device assignments and Winchester device names.

The swap device, /dev/swap, is assumed to be the first partition of drive 0, /dev/w0a. The "root" of the file system is assumed to be at block 8000, /dev/w0b. These notions are defined in the kernel, but the size and root base can be changed as detailed in section {12.7}.

If you boot from floppy, then the swap space and the root file system is on the floppy. The swap space on a floppy is not used, so it is only allocated a few sectors.

The following commands are available under System V.2:

diskconf /dev/w0h	Display drive format information
psize /dev/w0?	Display partition size

12.11 Multiple Winchester Drives

A multiple drive system can be used to increase storage capacity, improve performance or both. Increased performance is achieved by separating files and programs between drives so that average head seek time is minimized. One common configuration is to put the /tmp directory on a different drive from /usr, /lib and /bin, so the "C" compiler, link editor and similar programs will run faster.

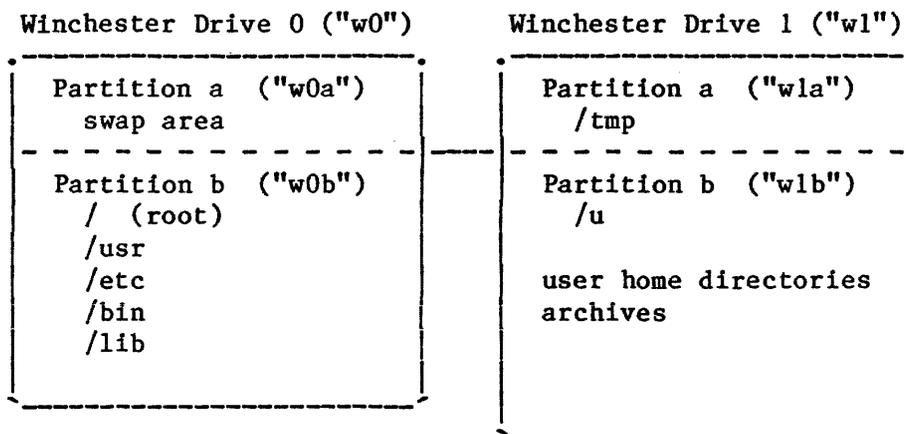


Figure 34. Multiple Drive Configuration - Typical

The drives do not have to be of the same type. Partition "a" on both drives, however, will be the same size since the swap size (nswap) is built into the kernel.

The `mount` or `df` commands will display the current configuration.

The general command sequence used to configure the system as shown above is:

```

diskformat /dev/rwlh ...
chstep /dev/rwlh 0 0
badblk /dev/rwlh
mkfs /dev/w1a ...
mkfs /dev/w1b ...
mkdir /u
mount /dev/w1a /tmp
mount /dev/w1b /u
  
```

Refer to section {12.6} and the `/usr/src/rebuild/rebuild` script for specifics on the `diskformat` and `mkfs` commands. The `mount` commands could be put in `/etc/rc` to be done automatically after booting.

13. ACCESSING I/O DEVICES (DEVICE DRIVERS)

There are a number of ways to access a physical I/O device. The following list ranks the methods from "worst" to "best". Each is described in detail below.

phys call: Not standard, Non-Portable.
Simple and quick to try.
Allows fast executing, in-line I/O.
No interrupt support.

/dev/mem: Somewhat better, but cumbersome.
Limited addressing range.
Slow access via kernel system calls.
No interrupt support.

Device driver: Most reliable and flexible.
Total device control, including interrupts.
Takes time and effort to implement.
Requires rebooting to test new version.

13.1 phys(2) System Call

The "phys" system call can be used to access an on-card or off-card I/O device. This call instructs the kernel to associate a particular range of virtual (logical) addresses with a particular range of physical addresses. Your program must be running as the super-user to execute this call.

```
phys(n,Vadrs,size,Padrs)
```

where "n" is 0,1,2 or 3
"Vadrs" is the base virtual address
"size" specifies the window size in bytes
(will convert to a power of two, 4096 or larger)
"Padrs" is the base physical address

```
Example: "phys(0,0x800000,4096,0xfe8000)"
```

assigns virtual addresses 0x800000 thru 0x800fff
to physical addresses 0xfe8000 thru 0xfe8fff.

Figure 35. `phys()` system call

The "jumper.c" program, below, uses the phys call to read the status of the user jumpers. (An easier way to access the jumpers and user LEDs is described in section {15.3}. This is just an example of phys usage.)

```

#include <stdio.h>
#define PSIZE 4096
#define JUMPHYS ((char *)0xfef000)
#define JUMPVIR ((char *) (0x800000-PSIZE))
main()
{
    if (phys(0,JUMPVIR,PSIZE,JUMPHYS)) {
        printf("must be super-user\n");
        exit(1);
    }
    for ( ;; ) /* do forever */
        printf("%x\r\n",*JUMPVIR);
}

```

Figure 36. `^jumper.c^` program (M10, V10)

The problem with the `phys` call method is that the user has no control over which virtual addresses are available for use and which are being used by the program, since they have been assigned by the operating system at run time. If you choose a value that is already in use by the kernel, there will be trouble when you reference that address, since a virtual address can only be mapped to one physical addresses. The critical factor is the value you choose for the virtual address.

Specifically, the `phys(2)` call will fail if the virtual address conflicts with some other address space in the program. Those other spaces are program text, data, stack, shared memory and other `phys` segments. In System V (M10, V10), text and data (combined) run upward from `0x000000`, and the stack grows downward from about `0xfefc00`. The virtual address of a shared memory segment is returned by the `shmat(2)` system call, and is usually between `0x800000` and `0xF00000`. System V.2 parameters are T.B.D.

A `phys(2)` call will also fail if the specified size, after rounding up to the next power of two, is not an exact multiple of the virtual or physical address arguments. This insures that only one MMU descriptor will be needed.

Unless your program is relatively simple, you should avoid the `phys(2)` call. If your memory environment changes, previously working `phys(2)` calls might fail due to changes in kernel-assigned virtual addresses. Also, remember that the `phys(2)` call may not be portable to other systems.

If you decide to use a `phys(2)` call, we suggest you start by trying the following virtual address values (M10, V10 systems):

- (a) `0x800000` minus your `phys(2)` segment size.
- (b) `0x800000`, if you are not using shared memory.
- (c) `0xff0000` (configuration dependent)
- (d) `"&end"`, rounded up to the next power of 2 boundary > 4096; see `ld(1)`.

Refer to section two of the UNIX User's Manual for more details on the phys call.

13.2 /dev/mem and /dev/kmem

The device `"/dev/mem"` allows you to access the first portion of physical memory as if it were an I/O device. In System V, this is `0x000000` through `0x7fffff`. You can place your I/O device near the end of that area; be sure it is not contiguous with RAM. Then, you can "open" `/dev/mem`, "lseek" to a particular byte position and "read" or "write" data. Refer to section two of the UNIX User's Manual for details on those system calls. Because of all the system calls required, this access method is somewhat slow and clumsy.

Device `"/dev/kmem"` allows you to access memory as the kernel sees it through the memory mapping logic (virtual memory). It is generally used in conjunction with `nlist(3)` and `adb(1)`. It should not be used to access physical memory.

On M10 and V10 systems, physical bus addresses above eight megabytes (`0x800000`) are inaccessible using `/dev/[k]mem`. See section {15.9} for more information on the memory configuration. System V.2 parameters are T.B.D.

13.3 Device Drivers - Reconfiguration Rights

This is the best way to handle devices. Unlike the methods discussed above, a UNIX device driver has complete control over a device and can handle interrupts. The only drawback is that you need considerable experience with UNIX and "C" in order to write one. There is little documentation available which will teach you how to write a driver. If you do not have any experience writing a device driver for UNIX, do not plan on whipping one up and having it work in a short period of time.

Follow these steps to learn how a device driver works and is hooked into the kernel:

- [1] Read "UNIX Implementation" in the UNIX Programming Guide.
- [2] Read "UNIX I/O System" in the UNIX Programming Guide. This describes the interface between I/O device drivers and the kernel.
- [3] For information concerning UNIX internals, including some discussion about the I/O system and device drivers, we recommend "The Design of the UNIX Operating System", by Maurice J. Bach, published by Prentice Hall.
- [4] Obtain "Reconfiguration Rights" from Heurikon. For system V, this is the source code for the existing device drivers and all files needed to make a new version of /unix. System V.2 source files are available "a la carte". Examine the files and use them as examples for your device driver. The actual content of recon-rights may change from those files listed below, as the distribution is updated from time to time.
- [5] Be sure you start with a simple driver first. Do one which has a minimum of fancy functions and interrupts. It may be possible to use an existing driver as the skeleton for a new one. Remember that a "character" device (such as the serial I/O driver) is easier to understand than a blocked device.

♣ Caution: You may violate a Copyright if you use an existing device driver as a skeleton for another driver unless you own complete rights to the original program. Don't let your editor lead you into trouble.

Reconfiguration Rights also allow you to modify certain system parameters, such as, the maximum number of active inodes, processes, open files and disk buffers. These values are initially set as a compromise between capacity and memory usage. You can use the 'pstat' command to see some of the current parameter usages and the maximum values.

The reconfiguration rights package does not include the source code for the kernel or utility programs. It only has the sources for files needed to modify or add device drivers and a binary copy of the kernel.

Reconfiguration rights also includes source code for the standalone bootstrap, if requested. Source code for the kernel is available separately; an AT&T license may be required. Contact us for details.

<u>File</u>	<u>Comment</u>
Makefile	For use with "make" to create /unix.
adb.script	Used during rebuild.
badints.c	Stubs for unused interrupts.
cio_util.c	Special pgms for CIO usage. (M10, V10, M220)
conf.c	Configuration info and driver hooks.
config.c	Initialization control for devices.
datasud_cent.c	Datasud Centronics driver. (V10)
datasud_ser.c	Datasud Serial driver. (V10)
dipled.c	User jumpers and LEDs driver. (M10)
dmainit.c	Initialization code for the DMAC. (M10, V10)
fd1793.c	SBX-FDIO Floppy disk driver. (M10)
fpp.c	68881 FPP device driver. (M10, V10)
it2190.c	Interphase 2190 SMD drive. (M10)
it3200.c	Interphase 3200 SMD driver. (V10, V20)
itc.c	Central Data Serial driver. (V20)
ivec.s	Interrupt vectors.
libex_driver.a	Ethernet driver, library.
linesw.c	Serial I/O line switch logic.
mb1031.c	CDC Serial Port Expansion board driver.
mfpser.c	MFP serial device driver. (V20)
mpc_msg.c	Message Passing (MPC) driver. (M220)
mt.c	Reel-to-reel tape driver. (M10)
mct.c	Reel-to-reel tape driver. (V10, V20)
mx.c	CP/M-Shell driver. (M10)
name.c	System version & compile time info.
omti_???.c	Drivers for Winch, Floppy & Streamer. (V20)
p3.c	Centronics printer driver. (M10)
plessey.c	Plessey Host adapter driver. (V20)
rtc.t	Real-Time Clock driver.
sbx.c	Initialization code for SBX modules. (M10)
scsi.c	SCSI Winchester driver. (M10, V10)
sccio.c	Serial I/O driver. (M10, V10, M220)
std.a	Device driver binaries, library.
streamer.c	Streamer tape driver. (M10)
unix.no_tp	UNIX kernel, w/o streamer, w/Centronics (M10)
unix.tp	UNIX kernel, with streamer tape.
unix.o	This is the UNIX kernel, in binary.
unix.f*	Various floppy-based versions.
windows.c	Windowing device driver.
VRTXio.o	VRTX R/T device driver (object).
xycom.c	Xycom Serial driver. (V10)

Table 7. Reconfiguration Rights - Contents (partial)

♣ Caution: The above listing is approximate. Actual files vary from release to release. Use this information for training purposes only.

Each "major" device number has a device driver associated with it. Basic sections of a device driver are:

- ✦ init. Called once at boot time from `config.c` to initialize the device.
- ✦ open(dev,flag). Called near the beginning of a program to prepare the device (`dev`) for an operation. For example, set the default baud rate. "Flag" is a modified version of the `oflag` value used in the `open(2)` system call; the bit assignments are defined in `file.h`.
- ✦ read(dev). Read data from minor device "dev". Used for a character device only.
- ✦ write(dev). Write data to minor device "dev". Used for a character device only.
- ✦ close(dev,flag). Do anything that is required after the last operation. For example, tape rewind.
- ✦ ioctl(dev,cmd,arg,mode). Do a special I/O device control function, as determined by "cmd". For example, change serial port baud rates or rewind a tape drive. The particular functions of the "ioctl" routine are device dependent. Used for a character device only.
- ✦ strategy(bp). Queues the requested I/O function for a block device and starts the I/O if the device is idle. The argument is a pointer to a buffer structure which contains details of the I/O request and the data.
- ✦ print(str,dev). A short routine which prints the string pointed to by "str", then expands and prints information about the device number, "dev", so as to identify the particular physical and logical drive. Used to print a kernel warning or error message concerning the device, e.g., if a bad block is found.
- ✦ Interrupt handlers. Usually the actual data transfer between the device and memory is done using DMA and interrupts. An interrupt handler services the device when it becomes ready and wakes the driver read/write routines when the transfer has been completed.

The following chart shows some of the relationships between the reconfiguration files and a device driver. The files "ivec.s" and "conf.c" are where the device driver is hooked into the kernel. Those files have interrupt vector tables and major-device-number switch tables, respectively.

A name with parentheses following it, such as "open()", represents a procedure name.

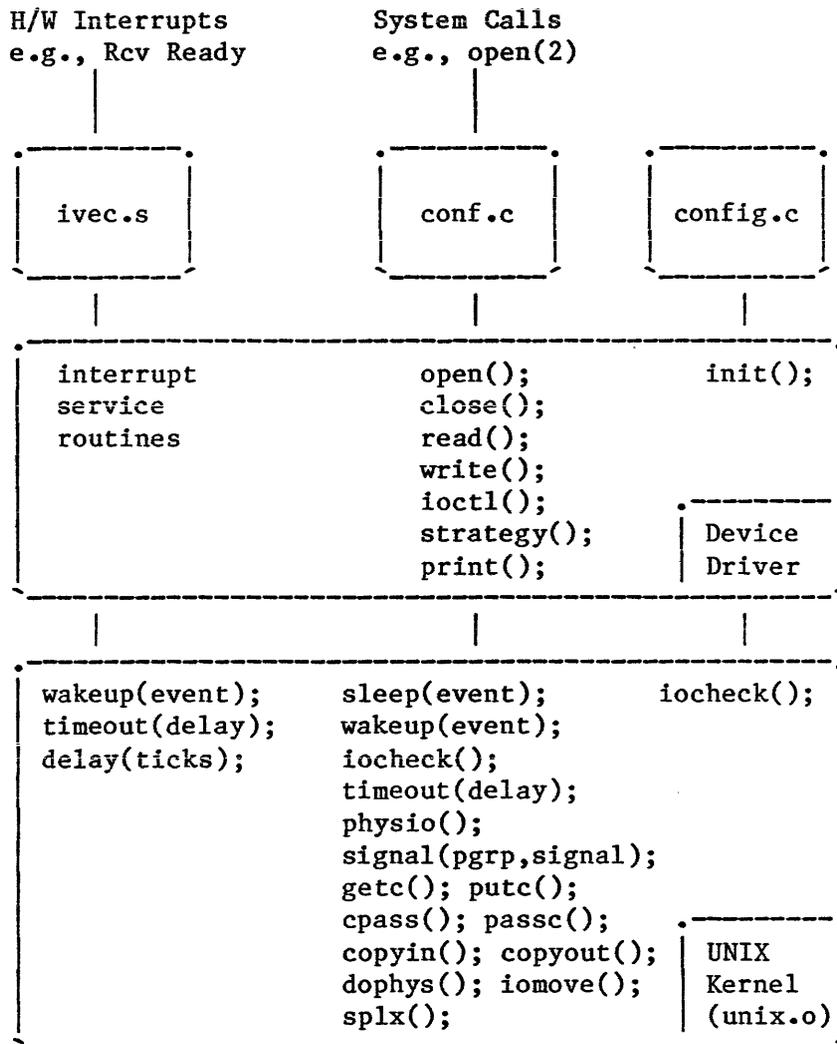


Figure 37. Device Driver and Kernel Hooks (Summary)

A blocked device is usually more complicated than a character device. There are more kernel routines used to schedule the device activities and control transfer of data blocks to and from the driver. Also, it is likely that the DMAC (direct memory access controller) is used, which somewhat complicates the driver logic.

13.4 Installing a New Device Driver

- ◆ Notice: this section is written using System V.0 as an example. In system V.2, files reside in subdirectories such as libconf and libdev. Also, the specific changes may vary from release to release. Use existing lines in the reconfiguration rights files as a guide for your additions or modifications.

Be sure to read the "UNIX I/O System" in the UNIX Programming Guide. The rest of this section will make more sense if you do that now. To wire-up the new device driver, a number of files must be changed.

- [1] First, connect the device driver entry points to the kernel. Edit `conf.c` as follows:

- a. Add a line of the form:

```
extern int devopen(), devclose(), devread();
extern int devwrite(), devioctl();
```

declaring the existence of the device driver entry points for use by the kernel. There are other similar lines in `conf.c`; place the new line at the appropriate spot just before the "cdevsw" or "bdevsw" arrays. The entry for a block device will have to declare two additional entry points, as follows:

```
extern int devstrategy(), devprint();
```

- b. For a character device, add (or replace) a line in the "cdevsw" structure of the form:

```
devopen, devclose, devread, devwrite, devioctl, 0,
```

in the "cdevsw" array. Each group of entries represents one major device number assignment, the first entry being major number zero. For a block device, the "bdevsw" entries are of the form:

```
devopen, devclose, devstrategy, devprint,
```

- [2] Next, connect the interrupt routines, if any, to the MPU interrupt logic. Examine the vector table, "`_dispatc`" in "`ivec.s`". This assembly language file causes the proper interrupt service routine (ISR) to be called, depending on the interrupt source. Each entry in the `ivec` table corresponds to one MPU exception vector, starting from zero. The CIO (or MFP on the HK68/V20) and bus interrupts are assigned the vector numbers listed in the MPU section of the HK68 User's Manual. If no interrupt is ever expected for a particular vector, a "jsr" to a routine in `badints.c` or a "jsr lfault" is used.

Edit `ivec.s` as follows:

- a. Replace one of the existing entries in "`_dispatc`" with a "jsr routine", where "routine" is a short setup routine at the end of `ivec.s`. Be sure you do not change the size of the dispatch table.
- b. Add an interrupt setup routine. For example:

```

        .globl  _isr
routine:movl  #_isr,sp@
        movw  #1,sp@-
        jmp   call

```

In this case, "isr" is the C language ISR; the compiler prepends an underscore to procedure names. The setup routine stacks two items: the address of the ISR and a 16-bit argument which is passed to the ISR. The argument allows one ISR to serve multiple interrupt sources. The "jmp call" goes off into the kernel to save the registers and status prior to calling the ISR. The stack is fixed so that the kernel gets control in order to restore the registers when the ISR returns.

- [3] (This step does not apply to HK68/V20 systems.) Edit "ciotable" in config.c, changing the CIO initialization values so that the proper interrupt is enabled. The table consists of pairs of numbers; the first item of each pair is a CIO register number and the second item is the value to be stored in that register. There are three sections to the table. The first group of values initializes CIO port A, which controls the on-card interrupts. The second group is for port B, which connects to the bus interrupts. The third group initializes timer three (to generate 60 Hz. time ticks) and enables the CIO as a whole. Examine the existing values and refer to the HK68 User's Manual and the Zilog CIO Technical Manual.
- [4] If the device needs initialization before UNIX starts or before interrupts are enabled, call the initialization routine from "oem7init" in config.c. This is a good place to do a facilities check if automatic recognition of the device is required (see "iocheck", below). Another method sometimes used to initialize a device is to do so on the first open. This method, however, is not appropriate if interrupts are used by the device.
- [5] Examine Makefile in reconfiguration rights. Edit Makefile so that the new driver will be compiled and included in the link edit phase. Usually, all that is required is that the driver name be added to one of the lines for "std.a". Test the new Makefile via `make -n unix.tp` (M10) or `make -n unix` (others).

13.5 Removing a Device Driver

Deleting a device driver is relatively simple. The only problems are usually the generation of undefined variable references, a situation which is corrected by searching for those references in all of the reconfiguration rights source files.

- [1] Remove the extern statements in conf.c which declare the driver entry points.

- [2] Remove the entries in the conf.c "cdevsw" (and "bdevsw") structures. Be careful to replace an entry with a dummy line of the form:

```
nodev, nodev, nodev, nodev, nodev, 0,    (for cdevsw)
nodev, nodev, nulldev, nulldev,         (for bdevsw)
```

If you do not do this, the major numbers for devices which have entries further down in the table will not be correct.

- [3] Remove any calls to the driver from config.c.
- [4] Edit ivec.s and ciotable to remove unused interrupt hooks.
- [5] Makefile may have to be changed, depending of the device removed.

13.6 Hints for Writing a New Device Driver

Here are some hints for writing and debugging a new driver.

- [1] Write the device driver, using some existing driver as a model. The serial driver (usually sccio.c) is a good character device example. Use the Winchester driver (usually scsi.c) for a block device example.
- [2] Use printf(3)'s liberally in the driver. Put them in the code at critical points, preferably not in loops. (The kernel printf routine temporarily suspends the entire system while the print is executed.) Bracket your print statements with #ifdef/#endif so that the debug messages can be turned off after the driver is finished. For example:

```
#ifdef DEBUG
    printf("new device: err=%d, ret=0x%x\n",errc,retn);
#endif
```

- [3] Keep interrupt service routines short and concise. Do as little processing as possible in an interrupt service routine. Interrupt stack space is limited, therefore, do not make excessive use of the stack in an ISR (i.e., use few automatic variables and don't use recursive routines).

13.7 Common Device Driver Problems

Here are some tips and common mistakes:

- [1] Your device driver must be re-entrant. You should assume, for example, that your read() routine will be used by more than one process at a time. Although, UNIX will not arbitrarily suspend one process in favor of another while executing at "kernel" level in a driver, another process may start up if you go to sleep. Use automatic variables, rather than globals, unless you really want a variable to be global.
- [2] An interrupt could occur and modify some parameter you are using. If you have a critical region, turn interrupts off, then back on.
- [3] Avoid large automatic arrays or structures in interrupt routines. Such items are placed on the interrupt stack, which is of limited size.
- [4] An interrupt service routine should not access user variables such as u_base or u_count. This is because the interrupt could occur at any time, not necessarily when a specific user is executing.

13.8 Kernel Routines and Macros

There are numerous kernel routines which a device driver can utilize. Many are described in the "UNIX I/O System" section of the UNIX Programming Guide. Some device drivers use kernel routines which are not documented; some of those are listed below.

- ♣ Use extreme caution when using any of these procedures; there is absolutely no guarantee that the characteristics of these routines will not change from release to release. These routines are not part of a defined interface. This listing is for training purposes only.

To learn more about these calls, examine the existing device drivers for examples of their use. Those marked "***" are discussed in the "UNIX I/O System" paper. Most of the macros used by a driver are defined at the beginning of the driver or in /usr/include/sys/sysmacros.h.

`iocheck(adrs);`

Tests if address "adrs" is accessible as a longword. Intercepts a bus error, and returns, instead of allowing the bus error to panic the kernel. Used for facilities checks using `setjmp(2)` during initialization. See `config.c` for examples.

```

    if ( !iocheck( (char *)0x700000 ) )
        device = NOT_PRESENT;    /* bus error, iocheck failed */
    else
        device = PRESENT;        /* iocheck passed */

```

`splx(s); **`

Sets the MPU interrupt level to "s" and returns the original level. Level zero is all interrupts enabled, level 7 is all interrupts except level seven disabled. Interrupts must be disabled during certain critical regions of code (e.g., to prevent race conditions or while making modifications to queue or buffer linkages; see "sleep", below).

```

    s_old = splx(s_new);

```

`printf(args);`

Similar to `printf(3)`, except for limited use within kernel. Supports a limited set of formats (`%c %s %u %d %o %x %D`). Caution: interrupts are turned off and all characters are output in polled mode while this statement is being executed. Use for debug or gross error messages only. Prints to the console only.

```

    printf("\nscsi: err %d, CIR=0x%x\n",err,reg);

```

`sleep(value,pri); **`

Suspend a process at priority "pri" until event "value" occurs (signaled via a wakeup). This call is usually done to wait for a physical event such as a device interrupt. "Value" is just a number which ties the sleep call to a corresponding wakeup call; usually an address value of some significant data item is used. The priority argument determines whether or not the sleeping process can be interrupted by a signal. `PZERO+1`, and greater, are interruptible sleep levels.

```

    SPL6(); /* prevent state from going TRUE between test and sleep */
    while ( state != TRUE ) /* wait for event, sleep may ret early */
        sleep(&queue,TTOPRI); /* does an internal SPL0() */
    SPL0(); /* back to normal */

```

Table 8. Kernel Routines and Macros (part 1)

Note: These table entries are in a random order.

```
wakeup(value); **
    Mark those processes sleeping on event "value" as ready to run.
    Usually called from an interrupt service routine. See sleep(), above.
    state = TRUE;
    wakeup(&queue); /* let sleeping process run */

timeout(proc,arg,delay); **
    Execute procedure "proc(arg)" after "delay" clock ticks. "Arg" is
    passed to the procedure as an argument. Timeout returns to the caller
    immediately. Do not use timeouts indiscriminately; the kernel will
    panic if there are too many (see NCALL in conf.c). A timeout, once
    set, cannot be canceled. A procedure can use timeout to restart
    itself.
    timeout(tmproc, dtrflag, HZ/2); /* start tmproc in 1/2 second */

delay(xx);
    Goes to sleep for "xx" clock ticks.
    delay(HZ*2);          /* disappear for two seconds */

signal(pgrp,sig);
    Send signal "sig" to all processes having the process group "pgrp".
    "Sig" corresponds to the signals listed in signal(2).
    signal(tp->t_pgrp,SIGKILL); /* kill the processes */

psignal(pid,sig);
    Send signal "sig" to process "pid".
    Sig" corresponds to the signals listed in signal(2).
    psignal(u.u_proc->p_pid,SIGFPE); /* send SIGFPE to the process */

prdev(str,dev);
    Prints the string, then calls the block device's print routine via the
    entry in bdevsw.
    prdev("fdstrategy: bad floppy format", bp->b_dev);

cpass(); **
    Returns the next character from the current process's output queue.
    Used by character devices to fetch the next character to send to the
    device. Usually found in the write routine of a character device.
    while( (c = cpass()) != -1 ) /* while characters to output */
        *DEVADDR = c;

passc(ch); **
    Places the character "ch" into the current process's input queue.
    Usually found in the read routine of a character device.
    while( passc(*DEVADDR) != -1 );
```

Table 9. Kernel Routines and Macros (part 2)

```
dophys(num,viradr,size,phyadr);
```

This is a driver callable entry point for the phys(2) system call. It bypasses the super-user check so that a device driver can map in an address region for a user program. The responsibility for security rests with the driver. It is typically used in the open routine (and the close routine to release the region). Refer to phys(2) and section {13.1} for argument and failure information.

```
    dophys(0, fppaddr, ctob(1), fppaddr);
    if ( !up->u_error ) {
        /* dophys failed */
    }
```

```
suword(adrs,value);
```

```
subyte(adrs,value);
```

Store the 32-bit (or 8-bit) "value" at "adrs" in the user's address space. These procedures compute the physical location of an address which is relative to the user's text and data space. They return -1 if "adrs" is out of range of the user's address space.

```
    if ( suword(addr, info) == -1 )
        up->u_error = EFAULT;
```

```
fuword(addr);
```

```
fubyte(addr);
```

Return the 32-bit (or 8-bit) value at "adrs" from the user's address space. These procedures compute the physical location of an address which is relative to the user's text and data space. They return -1 if "adrs" is out of range of the user's address space or, for fuword, if the target value itself is -1 (caution).

```
    if ( (info=fuword(addr)) == -1 )
        up->u_error = EFAULT;
```

```
copyin(src,dst,size);
```

Move "size" bytes from "src" in the user space to "dst" in the system (kernel) space. Returns non-zero if there is an error.

```
    if (copyin(addr, &steprate, sizeof(steprate)))
        u.u_error = EFAULT;
```

```
copyout(src,dst,size);
```

Move "size" bytes from "src" in the kernel to "dst" in the user space. Returns non-zero if there is an error.

```
    if (copyout(&data, addr, sizeof(data)))
        u.u_error = EFAULT;
```

Table 10. Kernel Routines and Macros (part 3)

```
iomove(addr,length,flag); **
    Similar to copyin and copyout, except automatically determines if data
    address is in the kernel or user space. Moves "length" bytes to or
    from u_base. The "flag" argument is either B_READ (move from addr to
    u_base) or B_WRITE (move from u_base to addr). This routine adjusts
    u.u_base, u.u_count and u.u_offset if addr is in user space and sets
    u_error on an error (but does not clear it if no error).
    iomove(cp, cc, B_WRITE);

iowait(bp);
    Suspends the process until the requested block I/O has been completed.
    This procedure executes a sleep. Sets u_error on an I/O error.
    iowait((struct buf *)bp);

iodone(bp); **
    Used by an interrupt routine to notify the kernel that the I/O for a
    particular block has been completed. This procedure does a wakeup.
    iodone((struct buf *)bp);

suser();
    Returns non-zero if the current process is executing as "root". This
    call can be used to enforce security by enabling certain functions
    only if the current user is the superuser.
    if ( !suser() ) {
        u.u_error = EPERM;
        return();
    }

disksort(iobuf, bp);
    Adds buffer "bp" to a list of buffers ready for I/O. Sorts the list
    so as to minimize head travel. Implements an elevator algorithm.
    disksort(&fdtab, bp);

physio(strategy,buf,dev,flag); **
    Used with a block device to implement a character device read and
    write. Physio calls the block device's strategy routine and waits for
    the I/O to be completed.
    physio(fdstrategy, &fdrbuf, dev, B_READ);

vtop(virt);
    Convert the virtual address "virt" in the user's address space to a
    physical address in system memory.
    physaddr = vtop(viradr);
```

Table 11. Kernel Routines and Macros (part 4)

```

getc(&queue); **
putc(c, &queue); **
    Manage character queues. A return of -1 indicates an empty (getc) or
    a full (putc) queue.
    while ( c=getc(&tp->t_outq)) >= 0 )
        todevice(c);

cmalloc(coremap, size, 0);                (M10, V10)
mfree(coremap, size, start);
    Allocate (or deallocate) memory for the current process's data area.
    Returns the click address of the new memory, or zero on error.
    "Coremap" is the name of the kernel's allocation table for memory.
    if ( buffer=ctob(cmalloc(coremap, btoc(sizeof(buf)), 0)) ) {
        /* okay, got the memory, use it... */
        mfree(coremap, btoc(sizeof(buf)), btoc(((int)buffer)));
    }

major(dev); **                            (MACRO)
minor(dev); **                            (MACRO)
    Returns the major (or minor) portion of the device number, "dev".

makedev(major,minor);                    (MACRO)
    Returns the device number, given the major and minor numbers. Thus,
    dev == makedev(major(dev),minor(dev)).

SPLx();                                  (MACRO)
    These macros set the MPU interrupt level. They are slightly less
    expensive (in time and space) than splx(). "x" is zero through seven.
    SPL0(); ..., SPL7();

ctob(clicks);                            (MACRO)
    This macro converts the argument, representing the number of memory
    blocks, to the size in bytes. A "click" is one memory allocation
    unit, or "block" (usually 4K bytes). It is based on a kernel
    compile-time constant.
    bytes = ctob(clicks);

btoc(bytes);                              (MACRO)
    The reverse of ctob().
    clicks = btoc(bytes);

btod(bytes);                             (MACRO)
dtob(blocks);                             (MACRO)
    Converts disk blocks to (or from) number of bytes (512 bytes/block).

```

Table 12. Kernel Routines and Macros (part 5)

13.9 Kernel Tables

The UNIX kernel makes heavy use of structures to describe a process or device state. Some of the most used structures and pointers are listed below.

- user** Defined in `user.h`. Contains information about a process which does not need to be referenced while the process is swapped out of memory. It has things like the user's register save area, effective user id, process timings and I/O pointers. It is referenced as "u.", as in `"u.u_error = EIO;"`. It is also called "U DOT". The area following the user structure, up to the end of the clik boundary, is used for an interrupt (supervisor) stack. The kernel always maps the U area of whichever process is currently running to the value of "&u". This simplifies references to the U area since the U area of all processes will be at the same (logical) address.
- proc** Defined in `proc.h`. Contains information about a process which must stay in memory when the process is swapped out. It contains the data such as the process's state, scheduler priority and memory mapping information. The `proc` structure is usually referenced using a pointer to the structure, as in `"pid = p->p_pid;"`.
- tty** Defined in `tty.h`. Contains information about a serial port, such as the line modes and pointers to the I/O queues. It is usually referenced using a pointer, as in `"tp->t_cflag = 0;"`.

14. I/O ERROR CODES14.1 Winchester Errors

If you get a Winchester disk I/O error, it usually indicates a problem with the data format on the drive. It could result from a severe power glitch or a physical problem inside the drive. If you have a persistent error, check the drive data and power cables and listen to hear if the drive is spinning properly. Contact our service department for assistance in debugging a drive error.

Controller Errors:

"Unable to select device"
 "Unable to select drive"
 "Request Sense returns bad status"
 "Non-zero completion message"

Drive Errors:

<u>type</u>	<u>code</u>	
0	0	No error detected
0	1	No index pulses from drive See if drive is spinning.
0	2	No seek complete from drive
0	3	Write fault from drive
0	4	Drive not ready after select Check cables and power.
0	5	Drive not selected - Check cables and power.
0	6	No track 0 from drive
0	7	Multiple drives selected
0	D	Seek in progress
1	0	ECC error in ID field
1	1	ECC error in data field
1	2	No Adrs mark detected on drive
1	3	No data mark detected
1	4	Sector not found
1	5	Seek error - Bad format
1	8	Correctable data error
1	9	Bad track flag set
1	A	Bad interleave code
1	C	Unable to read alternate track
1	E	Illegal alternate track access
2	0	Invalid command
2	1	Illegal disk access
2	3	Volume overflow
3	0	Controller internal RAM error

Table 13. SCSI I/O Errors

Note: This section does not apply to the Plessey interface (used on V20).

14.2 Reel-to-Reel Tape Errors (M10)

The list below pertains to unrecoverable reel-to-reel tape errors on an M10 system (using a Tapemaster controller).

<u>Code</u>	<u>Description</u>
1	Timeout waiting for Data Busy false.
2	Timeout waiting for Data Busy false, Formatter Busy false and Ready true.
3	Timeout waiting for Ready false.
4	Timeout waiting for Ready true.
5	Timeout waiting for Data Busy true.
6	System memory timeout.
7	Blank tape encountered, expected data.
8	Micro-diagnostic error.
9	Unexpected EOT or Load Point.
A	Unable to eliminate error on retry.
B	Read overflow or write underflow.
D	Drive interface parity error.
E	PROM checksum error.
F	Drive strobe timeout. Usually due to reading a record larger than written.
10	Tape not ready.
11	Tape write protected.
13	Missing diagnostic mode jumper.
14	Attempt to link unlinkable command.
15	Unexpected file mark encountered.
16	Parameter error. Usually due to a zero or too large of a byte count field.
18	Unidentifiable hardware error.
19	Streaming r/w operation terminated by the operating system or disk.

Table 14. Reel-to-Reel Tape Errors, Ciprico Tapemaster (M10 only)

Errors messages on V10 and V20 systems (which use the MCT controller and device driver) are self-explanatory.

14.3 Floppy Disk Errors

A SBX-FDIO (M10) floppy disk I/O error is described by a status byte, which has the following format:

<u>Bit</u>	<u>Meaning</u>
7(msb)	Drive not ready. Check diskette insertion and closed door. Check cables and power.
6	Write protect. The diskette is write protected.
5	Record type error.
4	Sector not found. Format error.
3	CRS error. Bad sector.
2	Lost data. DMAC error.
1	n/a
0(lsb)	n/a

Table 15. Floppy Errors (SBX-FDIO only)

14.4 Streamer Tape Errors

If a streamer tape error occurs, the error type is printed in English, rather than as a number.

Some of the most typical problems are:

- ♣ Tape cannot be written because it is write protected. Check the small round indicator in the corner of the cassette. In order to write, it should not be on "safe".
- ♣ Cassette not in place or inserted improperly. Remove and reinsert the cassette.
- ♣ A read error on a write protected tape or a rash of errors, which makes no sense, could be because the tape has been wound off the edge of the cassette. To avoid this problem in the first place, never remove a cassette while the tape is in motion and never manually crank the tape.

15. MISCELLANEOUS OTHER INFORMATION15.1 Floating Point Support

There are various methods which may be used to support floating point computations. They are summarized below.

- [1] Math Library. Use the standard math library routines, vi and math.h and libm.a. This requires no special hardware, but uses the most MPU time to perform computations.
- [2] 68881, In-line Code (M10, V10). The 68881 Floating Point Processor may be used on the M10 and V10 as a peripheral device. Special support software is available from Heurikon which uses the Green Hills compilers to generate 68010 instructions and in-line emulation of the 68881 coprocessor interface. The 68881 is installed on a M10 via a SBX module; the V10 has a 68881 socket.
- [3] 68881, Daughter Board (M10, V10). This is a special hardware module which plugs into the 68010 chip socket. The module has a 68020 MPU and a 68881 FPP. The 68881 operates as true coprocessor.
- [4] V20, 68881 Coprocessor (V20). The 68881 operates as a coprocessor via the on-card 68881 socket.

15.2 SUDH LEDs

Depending on the system, there are four or five status LEDs on the HK68 processor board. On some of our systems, these indicators are extended to the front panel. They continuously show the state of the HK68 MPU and DMAC.

<u>LED</u>	<u>Name</u>	<u>Meaning</u>
S	Supr	The MPU is in the Supervisor state.
U	User	The MPU is in the User state.
D	DMA	The DMAC has control.
B	Bus	The Bus is in control.
H	Halt	The HK68 has halted.

Table 16. HK68 Status LEDs

- S When the UNIX kernel is executing, the "S" LED will be on. This would be the case during I/O, any system call and when the system is idle.
- U The "U" LED indicates that a user program is executing. A program is said to be "CPU intensive" if the "U" LED is on heavily while it is executing.
- D When an I/O device is transferring a block of data, the "D" indicator will flicker. Data transfers are usually very fast, so this LED is hard to see.

- B If the HK68 is acting as a bus slave, this LED will flicker. This indicates that an intelligent I/O board is transferring a block of data.
- H You never want to see the "H" LED on, for if it is, the MPU has halted and only a hardware reset will unlock it. If the system halts with the "H" LED on, it usually indicates a hardware problem (e.g., bad memory), either on the HK68 or the bus.

15.3 User Jumpers and LEDs (M10)

Note: this section applies to HK68/M10 systems only.

There are eight software accessible user jumpers and eight user LEDs, which may be used to select options and indicate status conditions. These are currently used by some of the UNIX device drivers; however, they can also be reserved for the UNIX user.

The jumpers should normally be removed. From time-to-time, Heurikon uses various settings to debug portions of the device drivers, so you may see strange messages or the console display will slowly scroll, if any of these jumpers are installed. One LED normally indicates activity over the serial I/O ports. The LED will change state whenever a character is received or transmitted over any serial port. This is a useful indicator of activity with a remote site, a modem or any device not located near the system.

The jumpers and LEDs are accessible through device `"/dev/diplied"`. When the device is opened, the default action of the jumpers and LEDs is turned off. Reading from the device will return a byte containing the state of the eight jumpers, where a "1" bit indicates an installed jumper. Writing a character to the device will set the LEDs. Refer to `diplied(7)` at the end of this guide for details.

To manually turn the default actions of the jumpers and LEDs off, all that is necessary is to open the `/dev/diplied` port. For example,

```
touch /dev/diplied
```

will do it. You could include that command in `/etc/rc`, if you always wanted the jumper and LED defaults disabled.

You can (wastefully) "connect" the jumpers to the LEDs via:

```
cat /dev/diplied >/dev/diplied
```

An `^exterr /dev/diplied^` command will restore the default user jumpers and LED actions.

15.4 Using Environment Variables

Environment variables (which a program usually inherits from the shell) can be accessed from within a C program two ways. The hard way is to use the pointer passed as the third argument to `main()`. The easy way is to use code similar to the following fragment:

```
char *user;
if ( (user = getenv("USER")) == (char *)0L )
    user = getenv("LOGNAME"); /* no USER, try LOGNAME */
printf("name = %s\n", user ? user : "NONE" );
```

The `execle(2)` and `execve(2)` system calls may be used to create a new process and pass a specific environment.

15.5 /etc/update

`/etc/update` is a short program that executes the "sync" system call every 30 seconds, which flushes the disk buffers and the superblock to the Winchester. This insures that the Winchester is always up to date, just in case somebody trips over your power cord. `/etc/update` is started by `/etc/rc` when going to multi-user mode.

15.6 Sticky Bits and Shared Text

Certain programs can be "shared text" and have a "sticky" bit set. A program which is shared text can be executed concurrently by more than one user with only one copy required in memory. In addition, if the "sticky bit" is on, the executable image of the program will be retained on the swap device for faster loading. The "sh", "csh" and "vi" programs usually have their sticky bits set on.

To compile your own "sticky" programs, use the "-n" option with "cc" or "ld". That will make the program shared text. Use the `chmod` command to turn on the sticky bit, per the `ls(1)` and `chmod(1)` pages in the UNIX User's Manual.

There are some trade-offs to consider, however.

- ♣ Using shared text pgms will reduce RAM requirements if one program is needed simultaneously, multiple times.
- ♣ Using a shared text program with the sticky bit set will cause the program to load faster.
- ♣ Using a shared text program with the sticky bit set will consume swap space. If you run out of swap space, the system will crash.

To see if a program has been compiled with the "-n" option, use the `file` command. A "pure executable" file is shared text. The `pstat -x` command will show you the current swap space used by sticky programs.

15.7 Signals

Certain conditions, such as an illegal memory reference, will cause the system to abort a program or "panic". If the error occurs while a user program is executing, the kernel will simply abort the process, perhaps generating a "core" dump file. However, if the kernel was executing when the error occurs, it generally "panics". This is because, in a properly working system, the error is unexpected and there is no sensible or reliable method of recovery.

These conditions will cause the indicated signal to be sent to a user process (see "signal(2)") or, if the kernel is executing, a panic:

<u>Condition</u>	<u>Reason</u>	<u>Signal</u>
MMU Fault	Illegal memory access	SIGSEGV
Watchdog	No peripheral response	SIGSEGV
Adrs Error	Memory Alignment Error	SIGBUS
RAM Parity	Defective RAM chip	SIGBUS
DIV Zero	Divide by Zero	SIGFPE
Ill Instr	Illegal Instruction	SIGILL
Privilege	Privileged Instruction	SIGILL
Trap 1	Trap #1 (adb/sdb trace)	SIGTRAP
Trap 2	Trap #2	SIGIOT
Trap 3	Trap #3	SIGEMT
Spurious	No vector during INTACK	ignored

Table 17. Signals

Note that there are two causes for the SIGSEGV signal: either an access outside of the user's allocated memory area or a system bus timeout due to no bus acknowledge.

The system watchdog timer is used at boot time to size memory and do a facilities check (i.e., find out which I/O device are present). The watchdog is also used by the kernel to detect an access to an unassigned bus address. The watchdog timer must not be disabled.

The responses to these and other signals can be controlled by a user process, via the signal(2) system call.

15.8 RAM Considerations

A single user version of System V.0 or V.2 UNIX must have at least one megabyte of RAM for operation. The recommended minimum for System V.2, however, is two megabytes. The Green Hills Fortran and Pascal compilers (as well as the "C" compiler, in many cases) require two megabytes. Unless you have a special requirement, more than three or four megabytes of RAM will probably not be useful.

For system V (M10, V10), the maximum amount of RAM which will be recognized by UNIX is eight megabytes, less any space needed for memory

mapped I/O devices. The upper half of the memory map is reserved for special purposes and I/O devices. The UNIX RAM must be contiguous, starting at location 0. You can install non-contiguous RAM, but it will not be used directly by the kernel. If you want to prevent UNIX from using all of the available RAM, the "maxmem1" variable, defined in config.c, may be adjusted; its initial value is 0x800000. You can use `adb -k -w /unix` to read or modify maxmem1 in your /unix kernel file. Any change won't take effect until you reboot. If you have Recon Rights, maxmem1 can be changed in config.c; see section {13.3}.

System V.2 supports up to eight megabytes of RAM, in two non-contiguous areas (four megabytes on-card plus four megabytes off-card) Off-card expansion memory must be accessible over both the VME bus (for driver DMA hardware) and the VSB (for UNIX).

The amount of user memory (RAM, less kernel space) can be determined from the "mem =" message, printed at boot time, or by using adb to examine the long integer, "maxmem". The value in maxmem is in "clicks", which is the memory mapping segment size. There are 4096 bytes in a click.

If UNIX does not have enough memory for a program, it will "swap out" lower priority processes from RAM. Swapping is no problem, if it does not happen too often. However, if there is too big a demand on RAM, then swapping could start to consume a high percentage of the system time and performance will drop drastically. At its worst, a system which is doing excessive swapping is said to be "thrashing". This is the case where additional RAM will help. The "swapper" task (seen on a `ps -ef` output) should normally consume less than one percent of total system time.

With some programs (e.g., vi), you can tell if you are being swapped out by typing a character and observing if it is echoed to your display immediately or if there is a slight delay. A delay is due to your program being swapped back into memory. The `ps` and `pstat` commands can be used to see whether a program is in memory or swapped out.

A RAM parity error which occurs while a user program is executing will simply result in a "SIGBUS" signal being sent to the process. This will generally cause the process to terminate. If the kernel is executing when a parity error occurs, there is no graceful recovery, so the kernel "panics". A random parity error is a very rare occurrence; parity errors are more likely to signal a truly defective memory chip.

15.9 Memory Map

The UNIX system is configured with certain memory and I/O addresses reserved for optional features, such as Ethernet and serial expansion boards. The following diagrams show the approximate range of addresses currently in use. Since these values may change, be careful if you are installing your own peripheral cards. There is no guarantee that Heurikon will not allocate some of the unused space for future options. Refer to section {13} for more information on accessing I/O devices.

<u>Adrs</u>	<u>Memory Space (physical)</u>	<u>Notes</u>
FF FFFF	-----	(top of memory)
FF C220	-----	Floating Pt Processor
FF C200	SKY FFP	
FF 8000	-----	Reel-to-reel tape (V10)
	MCT	
FF 0100	-----	Reel-to-reel tape (M10)
	Tapemaster(00B0,00B1)	
FF 0000	-----	(base of bus I/O) MMU, DMAC, SCC, SCSI
	Ethernet(0010,0011)	
FE 0000	-----	Hbug monitor
	HK68 On-card I/O	
FC 0000	-----	(reserved)
	HK68 ROM (128K)	
F0 0000	-----	(shared memory and logical area for shared text.)
	(Twilight Zone)	
80 0000	-----	(top of /dev/mem)
7C 0000	-----	(CDC)
	(reserved)	
78 0000	-----	VRTX CP/M Shell
	8 Chnl serial Expn(1-4)	
70 0000	-----	(open)
	HK68 (1-15) MLZ-93 (1-4)	
x0 0000	-----	(maxmem1)
20 0000	-----	(optional) (top of 2 meg RAM)
10 0000	-----	(optional) (top of 1 meg RAM)
00 1000	-----	User RAM UNIX Kernel
	Exception Vectors	
00 0000	-----	(bottom of memory)

Figure 38. System V Physical Memory Map (M10, V10)

<u>Adrs</u>	<u>Memory Space (physical)</u>	<u>Notes</u>
FFFF,FFFF	-----	(top of memory)
E100,0000	-----	
	VRTX	(Also available for OEM use)
E000,0000	-----	
	(Twilight Zone)	

	RAM (bus)	(optional, 4 meg max)
0300,0000	-----	Start of Bus RAM (VSB & extended)
0240,0000	-----	(top of 4 meg on-card RAM)
	User RAM	
0210,0000	-----	(top of 1 meg on-card RAM)
	User RAM	
0200,1000	UNIX Kernel	
	Exception Vectors	
0200,0000	-----	Base of on-card RAM
01FE,0000	-----	
	Avail for OEM devices	
01FA,0000	-----	
	8 Chnl serial Expn(1-4) Ethernet	
01F0,0000	-----	
	RAM (optional)	(see below)
0100,0000	-----	Start of Standard Adrs
	Xycom Serial Interphase 3200 MCT	SMD Reel-to-reel tape
00FF,8000	-----	
	Avail for OEM devices	
00FF,4000	-----	
	Datasud Serial & Cent Plessey SCSI I/F	
00FF,0000	-----	Start of Short Adrs
	HK68 On-card I/O	MFP
00FE,0000	-----	
	(reserved)	

	HK68 ROM (128K)	Hbug monitor
0000,0000	-----	(bottom of memory)

Figure 39. System V.2 Physical Memory Map (V20)

For V20 systems, if on-card RAM size is one megabyte, then off-card RAM starts at VME bus address 0110,0000, with VSB access at 0300,0000. If

on-card RAM size is four megabytes, then off-card RAM starts at VME bus address 0140,0000, with VSB access at 0300,0000.

15.10 Interrupt Usage

Bus interrupts are used for signals to or from various device controllers. The exact allocations are system dependent, however, here is one typical configuration:

<u>Device</u>	<u>M10 Bus</u>	<u>V10 Bus</u>	<u>V20 Bus</u>
EXOS Ethernet	5	4	5 (vectored)
Tapemaster Reel-to-Reel Tape	3		
MCT Reel-to-Reel Tape		5	T.B.D.
CDC MB1031 Serial Expansion	2	2	2
IT 2190 SMD Interface	1		
IT 3200 SMD Interface		3	3 (vectored)
Plessey (OMTI interface)			3 (vectored)
Datasud (Centronics, Serial)			1 (vectored)

Mailbox interrupts are also used, to extend the capability of the I/O devices.

By the way, a "mbi7" message from the kernel indicates that a CIO interrupt input went false before the MPU interrupt acknowledge cycle started. You might see this while debugging a new (buggy) device driver or if there is an unterminated bus interrupt line.

15.11 DMAC Channel Assignments

(This section does not apply to the V20)

<u>Channel</u>	<u>M10</u>	<u>V10</u>
0	SCSI	SCSI
1	Streamer (P3)	n/a
2	n/a	n/a
3	SBX-FDIO	n/a

Table 18. DMAC Channel Assignments

"n/a" means not assigned.

15.12 Making pROMs

If you compile a program with the intent of putting the code in a read only memory (ROM), then you must observe these rules. Note: if you are System V.2, use the cc5.0, as5.0 and ld5.0 utilities.

- [1] You cannot initialize external and static variables in a declaration; they must be explicitly initialized in your program. This is because the program will not be loaded and initialized by the kernel, as would be the case if you were executing under UNIX.
- [2] You cannot rely on variables being initialized to zero, as is the case when running under UNIX. Variables will start with an indeterminate value. Clear "_edata" to "_end".
- [3] Of course, you cannot use the UNIX system calls or any library subroutines which make a kernel call.
- [4] Include a short machine code (asm) program to initialize your stack pointer and branch to your "C" code ("_main"). Don't forget to include the proper RAM "turn-on" code, as explained in the HK68 User's Manual.
- [5] Use the -OPS and -c options with "cc" to prevent stack probe instructions from being inserted and to prevent the link edit phase from being done automatically when you compile. The -c option is necessary to prevent the standard C startup routines from being attached to your program.
- [6] The following link editor command will force the ROM and RAM portions of your program to be at the addresses you specify. (Use ld5.0 for System V.2). In this example, the ROM code is placed at 0xFE0000 and the variables are at 0x400. Your data must be contiguous with your text areas; do not use the LD option.

```
cc -c file1.c
ld -LT FC0000 -LC 400 file1.o [file2.o ...] -o ld.out
```

- [7] Use `nm ld.out` to verify the locations of variables and constants.
- [8] The ld.out file can be sent to the pROM programmer using a custom program to output in the programmer's format. The "hex" command can be used if your programmer understands "S" records.

```
hex ld.out > hex.out
```

Other commands which may help are "od", "see" and "cat". You can modify the format of the hex file using the stream editor, "sed". It may be necessary for you to separate your even and odd bytes if your programmer cannot do so internally.

- [9] If you output to your programmer over a serial port, you may need to put the port to "sleep" and use the "stty" command to set the baud rate and other modes. Refer to sections {16.7} and {16.10}.

15.13 System V.2 Porting to/from System V.0

In general, programs compiled under System V.0 (on an M10 or V10 system) will run on a System V.2 (V20), although shared text binaries are not interchangeable. Due to certain quirks, such as not having access to the 68881 FPP coprocessor or the symbolic debugger, we recommend that when both systems are available, all programs be developed under System V.2 with V.2 utilities. If pROMS need to be made, the System V.0 utilities (e.g., ld5.0) should be used.

Some other considerations are:

- ✦ Valid cc5.0 flags are "c", "O", "o" and "r".
- ✦ Avoid use of cc5.0 flags "s" and "n".
- ✦ To produce a shared text program on V.2 to run on V.2:

```
cc -o outfile -O -s -n infile.c
```

or

```
cc -c -O file1.c
cc -c -O file2.c
cc -o outfile -s -n file1.o file2.o
```

- ✦ To produce a shared text program on a V.2 system to run on V.0:

```
cc5.0 -o tempfile -O -r infile.c      (on V.2)
ld -o outfile -n -s tempfile         (on V.0)
```

or

```
cc5.0 -c -O file1.c                  (on V.2)
cc5.0 -c -O file2.c                  (on V.2)
cc5.0 -o tempfile -r file1.o file2.o (on V.2)
ld -o outfile -n -s tempfile         (on V.0)
```

- ✦ To produce a shared text program on a V.0 system to run on V.0:

```
ld -o outfile -n -s infile
```

- ✦ To produce a shared text program on a V.0 system to run on V.2:

```
cc -r -o tempfile infile.c          (on V.0)
cc5.0 -o outfile -n -s tempfile     (on V.2)
```

- ✦ Use `^coffbin -s -u file^` to convert a V.2 "coff" format file to a V.0 binary. This command operates on "file"; the named file is altered.

15.14 Shared Memory, Semaphore and Messages

The UNIX manuals do not include examples showing how to use the shared memory, semaphore and message features of the system. The introduction to section two, `intro(2)`, details the various structures which are used, and the various system calls are documented in section two.

- ✦ Shared Memory is used when two or more processes need to access a common area of memory, each being allowed, perhaps, to read or write the data stored there. This permits separate processes to exchange data or control information. Shared memory segments may even remain allocated when no processes currently exist which are using the data. The processes must have a common understanding of the data structure, generally accomplished via a ".h" file.
- ✦ Semaphores are used to provide exclusive use of some resource, perhaps a memory region or a device. A process may read and set the semaphore "value". Usually, a zero value indicates that no process "owns" the semaphore, while a non-zero value connotes ownership or some status information. The kernel can be instructed to suspend a process until the semaphore is released by another process.
- ✦ Messages allow programs to communicate with by sending a (usually short) block of data to each other via the kernel. Processes can "send" and "receive" messages via queues maintained by the kernel.

The following program shows examples of shared memory and semaphore system calls. The message passing system calls are used in a similar fashion. This program doesn't really do much; its only purpose is to illustrate a typical sequence of system calls needed to allocate shared memory and grab semaphores. This program creates, uses and removes a shared memory segment and a semaphore. Since those items are used only by this program (and its children), the "key" value is "IPC_PRIVATE". If you have a number of separate programs which need to access these features, the programs must have a prior agreement on the key value.

You can monitor the current status of all semaphores, shared memory segments and messages in the system by using the "ipcs(1)" command. Garbage items can be removed via the "ipcrm(1)" command.

Be aware that these resources might remain allocated even after all programs using them exit. This could cause problems later, such as running out of user memory for executing programs. When you allocate the resources, you can specify that they should be removed when all programs exit, or you can manually remove them via the `ipcrm(1)` command.

```

#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <sys/sem.h>
#include <errno.h>
#include <stdio.h>
#include <sys/signal.h>

int    semid = -1;        /* semaphore id */
struct sembuf acquire[2]; /* structures to get semaphore */
struct sembuf release[1]; /* structure to release semaphore */
struct shared_memory {   /* example shared memory structure */
    short  item1;
    short  item2;
};
#define SH_SIZE sizeof(struct shared_memory)
int    shmid = -1;       /* shared memory segment id */
struct shared_memory *shm; /* pointer to shared memory segment */
#define NCHILD 2
int    child[NCHILD];

main()
{
    int cleanup(), i, number;
    signal(SIGTERM,cleanup); /* don't leave a mess when done */
    signal(SIGINT,cleanup);
    if ( (shmid=shmget(IPC_PRIVATE,SH_SIZE,SHM_R|SHM_W)) < 0 ) {
        perror("xx: shmget fail");
        cleanup(0); /* we don't return */
    }
    if ( (shm=(struct shared_memory *)shmat(shmid,NULL,0)) < 0 ) {
        perror("xx: shmat fail");
        cleanup(0);
    }
    /* create a semaphore... */
    if ( (semid=semget(IPC_PRIVATE,1,SEM_R|SEM_A)) < 0 ) {
        perror("xx: semget fail");
        cleanup(0);
    }
    /* setup semaphore structures for use by slock and sunlock */
    release[0].sem_op = -1; /* decrem semval */
    release[0].sem_num = 0;
    release[0].sem_flg = 0666;
    acquire[0].sem_op = 0; /* if semval!=0, suspends */
    acquire[0].sem_num = 0;
    acquire[0].sem_flg = 0666;
    acquire[1].sem_op = 1; /* add one to semval */
    acquire[1].sem_num = 0;
    acquire[1].sem_flg = 0666;
}

```

```

for ( i=0; i<NCHILD; i++ ) {
    if ( (child[i]=fork() ) == 0 ) { /* clone thyself */
        signal(SIGTERM,SIG_IGN); signal(SIGINT,SIG_IGN);
        while ( 1 ) { /* this is a child */
            shm->item1 = shm->item2 + child[i];/* access shmem */
            if ( slock() == getpid() ) { /* lock semaphore */
                /* do work here, this process had it last */
            } else {
                /* work here, this process did not have it last */
            }
            sunlock(); /* release semaphore */
        }
    } else if ( child[i] < 0 ) /* fork error */
        perror("xx: fork error");
}
while ( 1 ) /* this is the parent */
    shm->item2 = number++; /* example shared memory access */
}
slock() /* get the semaphore */
{
    int pid, ret;
    if ( (pid=semctl(semid,0,GETPID)) < 0 )
        perror("xx: semctl fail");
    /* suspend and get semaphore */
    while ( (ret=semop(semid,acquire,2)) < 0 && errno == EINTR )
        if ( ret < 0 )
            perror("xx: semop, acquire fail");
    return(pid);
}
sunlock() /* release semaphore */
{
    if ( semop(semid,release,1) < 0 )
        perror("xx: semop, release fail");
}
cleanup(sig)
{
    int i;
    for ( i=0; i<NCHILD; i++ )
        if ( child[i] )
            kill(child[i],SIGTERM);
    if ( semid >= 0 )
        if ( semctl(semid,0,IPC_RMID,0) != 0 ) /* remove semaphore */
            perror("xx: semctl IPC_RMID fail");
    if ( shmids >= 0 ) { /* remove the shared memory segment */
        if ( shmdt(shm) != 0 )
            perror("xx: shmdt fail");
        if ( shmctl(shmid,IPC_RMID,0) != 0 )
            perror("xx: shmctl fail");
    }
    exit(0);
}

```

Figure 3. Semaphore and Shared Memory Example Program

15.15 Accessing Kernel Variables

The `nlist(3)` subroutine allows kernel variables to be accessed from a user program. This is the facility used by `'ps'` and other system utilities to fetch and display kernel variables.

The following program illustrates the use of the `nlist(3)` routine.

```

/* probe the depths of the kernel */
#include "fcntl.h"
#include "stdio.h"
#include "sys/types.h"
#include "a.out.h"      /* needed by nlist(3) */
#include "sys/sysinfo.h" /* for the example, below */

#define fetch(index,thing) (lseekit(fdmem,(long)value(index), 0),\
                           readit(fdmem, &thing, sizeof(thing)))
#define getval(adrs,thing) (lseekit(fdmem,(long)adrs, 0),\
                           readit(fdmem, &thing, sizeof(thing)))
#define value(index)      (nnL[index].n_value)

struct nlist nnL[] = {
    { " lbolt" },      /* lbolt is total ticks since boot */
    { " _sysinfo" },  /* sysinfo is start of system info table */
    { 0 }},           /* marker, end of table */
};
#define LBOLT          0      /* index into nnL[] */
#define SYSINFO        1

time_t lbolt;          /* a place to put the value when we read it */
struct sysinfo *psysinfo; /* pointer to data in the kernel ... */
struct sysinfo sysinfo; /* and a place to put the data when read */

char *pgmname; /* for use with perror() */

main(argc,argv)
char **argv;
{
    int i, fdmem;

    pgmname = argv[0]; /* save program name for perror() */
    if ((fdmem = open("/dev/kmem",0)) < 0) { /* open memory */
        perror(pgmname);
        fprintf(stderr, "%s: cannot open /dev/kmem\n",pgmname);
        exit(1);
    }
}

```

```

if (close(open("/unix", 0)) < 0) { /* test availability for nlist() */
    fprintf(stderr, "%s: cannot open /unix\n",pgmname");
    exit(1);
}
if ( nlist("/unix", nnL) != 0 ) { /* read namelist from kernel */
    fprintf(stderr, "%s: bad nlist\n",pgmname");
    exit(1);
}
for ( i=0; nnL[i].n_name[0]!=0; i++) { /* validate nlist structure */
    if (nnL[i].n_type == 0) {
        fprintf(stderr, "%s: incomplete namelist in /unix\n",pgmname");
        exit(1);
    }
}
/* EXAMPLE: print a nlist value */
printf("&lbolt = 0x%x (adrs)\n",value(LBOLT));

/* EXAMPLE: simple, one-level fetch and print
   (nlist gave us a pointer to a value) */
fetch(LBOLT,lbolt); /* get the value from the kernel */
printf("lbolt = %d (ticks)\n",lbolt);

/* EXAMPLE: two-level fetch and print
   (nlist gave us a pointer to a structure) */
psysinfo = (struct sysinfo *)value(SYSINFO); /* get the pointer */
getval(&(psysinfo->rcvint), sysinfo.rcvint); /* get the value */
printf("sysinfo.rcvint = %d (interrupts)\n", sysinfo.rcvint);
}

lseekit(fd, offset, whence) /* lseek with error checking */
long    offset;
{
    if (lseek(fd, offset, whence) == -1) {
        perror(pgmname);
        fprintf(stderr, "%s: error on lseek\n",pgmname");
        exit(1);
    }
}

readit(fd, buf, nbytes) /* read with error checking */
char    *buf;
{
    if (read(fd, buf, nbytes) != nbytes) {
        perror(pgmname);
        fprintf(stderr, "%s: error on read\n",pgmname");
        exit(1);
    }
}
}

```

Figure 41. nlist(3) example

15.16 System V.2 Notes

On a V.2 system, the following programs are in /usr/local/bin, which is first in the shell's search path. These programs are additions to or modifications of the standard UNIX release.

- adb This is the system V.0 debugger. It is supplied with system V.2 to allow conversion of a floppy kernel to a disk kernel during the rebuild process. Adb is not supported under system V.2 (see `sdb`).
- badblk V20 Badblk program. `badblk /dev/rw0h`
- captainfo Takes a single termcap entry from standard input and translates it to terminfo input on standard output. Sources are in /usr/local/src.
- captainfo <file.ltcap> file.linfo
 tic file.linfo
- chktrks Simply checks the disk format track by track. This will not damage data already on the disk. Badblk will check and allow you to map out badblks. `chktrks /dev/rw0h`
- chstep Moved here to distinguish it as System V.2 for rebuild script.
- coffbin Coff format to a.out format conversion program. Used to allow adb to redo the floppy kernel at rebuild time. It rewrites the filename given to it. `coffbin -u -s filename`
- cpio This version has the -F option which causes the blocking factor for tape to be 64. This makes reading and writing of streamer tapes much faster.
- diskconf Checks the disk parameters. `diskconf </dev/rw0h`
- ld5.0 Has symbol table increased to about 10,000 symbols.
- psize Displays the size of the disk partition. `psize </dev/rw0b`
- status See below.
- statwrite Status and statwrite are diagnostic programs for the Central Data smart serial cards.
- stretension Streamer retension program.
- tar This version has the `b` option which allows blocking factors as high as 64. As in cpio, above, this improves streamer tape performance.

- untic Converts a terminfo entry into ASCII form, allowing changes. The modified ASCII file is put back via the tic command. Sources are in /usr/local/src.
- vmail, xmail Allows mail using the vi editor. Not supported. Alias mail to vmail and check that xmail is linked or copied to /usr/bin/xmail.

16. SERIAL PORT CONFIGURATION

Heurikon HK68 microcomputers have from one to four on-card serial ports. On some boards, up to eight expansion ports may be added to the HK68 via plug-on SBX modules, four additional ports per module. In addition, one to four, eight-channel serial expansion boards may also be added to the bus. These may be used instead of the SBX modules or in addition to them. There are three types of serial ports on the Heurikon UNIX system. These are summarized below:

- ♣ **Standard Ports:** These are conventional serial ports which would be used for connecting to devices such as terminals and printers.
- ♣ **Modem Ports:** These ports are designed for auto-answer modems. They monitor certain control lines from the modems to determine if a call has been answered or terminated.
- ♣ **Network Ports:** These ports allow an interconnection between two systems. They are configured so that one RS-232-C cable can be used to permit concurrent communications between two Heurikon UNIX M10 systems, with either system acting as a master.

Each of these port types is discussed in more detail below, along with the necessary information which you would need to properly configure your system and files to use the serial ports.

16.1 Standard Ports

Standard ports connect to simple devices such as terminals, printers and originate modems. The "console" device is assumed to be connected to the on-card HK68 port "B". The other ports may be used as desired. These ports have names such as:

```
console (or usually simply noted as "co")
tty0
tty1
(etc)
```

The console device is unique in that certain system messages will always be directed there. (See sections {10.10} for information on /dev/syscon). Some installations use a hard copy terminal for the console in order to keep activity and error logs. We recommend that you simply use a CRT for the console, assign it to the primary person responsible for the system and leave it on at all times.

16.2 Modem Ports

Modem ports differ from standard ports in that the software monitors certain modem control lines to determine if an incoming call has been answered and if the call is still in progress. If a connection is terminated during a session (loss of carrier), the "hangup" signal will be

sent to your shell, which will log you off. These ports should only be used for modems which are intended to receive calls.

The following modem signals are used: (For a complete listing of the interface signals and their functions, refer to section {16.4}, below.)

DSR	Data Set Ready. ("D" modem pin 6, HK68 pin 4)	from modem
RI	Ring Indicator. ("D" modem and HK68 pin 22)	from modem
DCD	Data Carrier Detect ("D" modem pin 8)	from modem

There would normally be a "getty" running on a modem port, waiting for an incoming call. The getty starts by initializing the port, which causes DTR and RTS to turn on, and then it goes to "sleep" waiting for the modem to indicate that a call has been answered. When the modem brings DSR true, the getty process is given a "wakeup" to start the login process. For the on-card ports (four on M10, two on V10), both DSR and RI must be true for the wakeup call to be issued. The eight channel CDC serial expansion boards (on the bus) require both DSR and DCD to be true. This allows "intelligent" modems to be used, such as the Ven-Tel-212, which keeps DSR true between calls.

On the VMEbus Datasud serial expansion board (two ports), only DSR (pin 6 or 20, depending on Datasud jumpers) is used to control the getty.

Once a call has been answered, the modem maintains DSR (and RI or DCD on an M10 or CDC) in the true state. When the connection is broken, either by loss of carrier (phone line) or removal of DTR (UNIX), the modem will drop DSR. This will cause a hangup signal (logout) to be sent to the shell attached to the modem port.

During the session, the modem must maintain DSR true. In addition, if you are using a M10 or V10 on-card port, RI must also stay true, as is the case with the Ven-Tel 212 series of modems. If your modem does not maintain RI, then you should either use a special interface cable or an off-card port, which do not monitor RI.

Modem ports have names such as:

tty2m tty3m (etc)

If you are using a modem for an outgoing call, use a standard port, such as "tty2". The DTR signal from the HK68 will be on as long as the port is open (active).

If you are using an intelligent modem and wish to be able to receive calls and make outgoing calls over the same port, configure the system as follows: (tty2 is used as an example.)

```
device tty2m:  getty (/etc/inittab)
device tty2:   use for outgoing calls
               ("cu" or "uucp")
```

On System V.0, this bi-directional feature is available only on SBX-SCC, Datasud and on-card ports; not on CDC or other expansion ports. On System V.2, Datasud and CDC expansion boards support the modem logic.

When the tty device (e.g., tty2) is opened for the outgoing call, the getty running on the corresponding modem device (tty2m) will be automatically killed and the modem may then be used for the outgoing call. When the call is completed, a new getty will be allowed to monitor the modem for an incoming call. If the modem is "active" with an incoming call, the attempt to open the tty device will fail. This feature allows a single port and modem combination to be used for both incoming or outgoing calls.

If your modem port is wired and configured properly for auto answer, a `ps -e` should normally display a "?" in the TTY column for the modem getty. If, for example, you see "2m", the system thinks an incoming call is in progress.

Avoid using automatic baud rate selection logic of getty or your modem. Automatic baud rate selection based on the first characters coming to or going from the CPU is not very reliable.

By the way, if you hook your system to the phone lines, check that every line in your `/etc/passwd` file has a non-null password field.

Various makes of modems may be used with the Heurikon UNIX system (see section {16.8.1} for a list). The following sections detail wiring and configuration information for some specific types. Contact us if you need help connecting another variety.

- ♦ The big secret in getting a modem to work properly is to use a breakout box to look at the signals and permit wiring experiments.

16.2.1 Ven-tel 212-4 (ACUVENTEL)

<u>Switch</u>	<u>ON</u>	<u>OFF</u>
S6	2,4,5,6,7,9	1,3,8,10
S7	1,8,10	2,3,4,5,6,7,9

Table 19. Ven-Tel 212-4 Modem Switch Settings

Wiring is per section {16.4}. When you use a Ven-Tel modem, be sure you have the proper end of the telephone cable plugged into the modem. The modem end should only have two of the four wires attached in the RJ11 jack. Look closely. If this is incorrect, noise on the extra two wires

can cause the modem to capture and busy-out the phone circuit.

The Ven-tel modem will not work in auto-answer mode on a CDC expansion port. It won't terminate the session when you hang up because of the RI and DCD signal characteristics.

16.2.2 Hayes Smartmodem 1200 (ACUHAYES)

Switch	<u>UP(off)</u>	<u>DOWN(on)</u>
	1,2,4,5,6,7	3,8
	9,10 (if present)	

Table 20. Hayes Smartmodem 1200 Switch Settings

<u>HK68</u>	<u>MODEM</u>	
pin 2	pin 3	Data from modem
pin 3	pin 2	Data to modem
pin 4	pin 6	DSR
pin 5	pin 20	DTR
pin 7	pin 7	Ground
pin 20	pin 5	CTS
pin 22	pin 8	DCD

Table 21. Hayes Smartmodem 1200 Wiring

16.2.3 US Robotics Password Modem (ACUUSR)

Switch	<u>ON</u>	<u>OFF</u>
	(none)	1,2,3,4

Table 22. US Robotics Password Modem Switch Settings

<u>HK68</u>	<u>MODEM</u>	
pin 2	pin 3	Data from modem
pin 3	pin 2	Data to modem
pin 4	pin 6	DSR
pin 5	pin 20	DTR
pin 7	pin 7	Ground
pin 20	pin 5	CTS
pin 22	pin 8	DCD

Table 23. US Robotics Password Modem Wiring

16.2.4 Novation Professional 2400 Modem (ACUNOVATION)

<u>Switch</u>	<u>ON</u>	<u>OFF</u>
A	2,4,5,8	1,3,6,7
B	3,7	1,2,4,5,6,8
C	1	2,3

Table 24. Novation Modem Switch Settings

<u>HK68</u>	<u>MODEM</u>	
pin 2	pin 3	Data from modem
pin 3	pin 2	Data to modem
pin 4	pin 6	DSR
pin 5	pin 20	DTR
pin 7	pin 7	Ground
pin 20	pin 5	CTS
pin 22	pin 8	DCD

Table 25. Novation Modem Wiring

16.3 Network Ports

The term "network", as used here, means an interconnection between two Heurikon M10 computer systems. When two Heurikon M10 UNIX systems are connected via a serial link, usually one system acts as a master and the other as a slave. The slave system has a getty running, which waits for the master to login and initiate commands. These commands may cause processes to be started on the slave machine or data to be transferred between systems. When the master is finished, it does a logout. The Heurikon serial network ports allow either machine to act as a master. The single physical serial link has two logical devices at each end, referred to as device "a" and device "b". On system 1, any characters associated with the "a" side will correspond with the "b" logic on system 2, and vice versa. This results in two independent communication "channels" using only one serial link.

System 1 port "a" connects to System 2 port "b".
System 1 port "b" connects to System 2 port "a".

The physical connection is shared by the two channels. Characters received at each end are automatically routed to the proper logical device (i.e., the "a" or the "b" side) by the serial device driver. Channel priority is determined on a first come, first served basis, except that neither port is allowed to "hog" the interface for more than 20 consecutive characters.

This logical connection method allows each end of the network to be configured the same, e.g., gettys running on the "a" side at both ends. Either system may then act as a master and initiate a conversation on the "b" side of the link via "cu" or "uucp".

The serial network ports have names such as:

```
tty3a
tty3b
tty4a
tty4b
```

Serial ports which are configured for "network" operation cannot be used as a standard port. That is, if "tty3a" and/or "tty3b" are open, you will not be allowed to access "tty3". If you try to do so, you will receive a "Mount Device Busy" or "device unavailable" message.

The default baud rate for the network ports is 4800. This value cannot be changed via "stty". It has been chosen so that the probability of overrunning the serial receiver is low.

The standard version of the Heurikon UNIX system allows network ports to be assigned to the last two on-card devices and the first two SBX-SCC expansion ports on P7 (four total). These would normally be tty devices 1a, 1b, 2a, 2b, 3a, 3b, 4a and 4b.

16.4 RS-232-C Connections

16.4.1 HK68 and SBX-SCC

This section lists the RS-232-C signals which are used by the HK68 and a serial device. Except for the "RI" signal, this section applies to both the HK68 and the four channel SBX-SCC expansion modules. See section {16.4.2} for information on the eight channel expansion board.

The HK68 is designed to appear as a "data set" at the interface. When a "data terminal" type device (e.g., a CRT terminal or a printer) is to be connected, a direct hook-up may be used.

HK68 <u>Signal</u>	"D" pin <u>Number</u>	<u>Dir</u>	"D" pin <u>Number</u>	Terminal <u>Signal</u>
Rcv Data	2	<-	2	Tx Data
Tx Data	3	->	3	Rcv Data
DCD	4	<-	4	RTS
DTR	5	->	5	CTS
RTS	6	->	6	DSR
Gnd	7		7	Gnd
CTS	20	<-	20	DTR

Table 26. Connection to a Data Terminal Device

Notice that the signals go straight through, although their names depend on which end of the interface you're on. D pin 20 is not used on the HK68/V20.

When a "data set" device (such as a modem) is connected, a cable reversal must be inserted between the HK68 and the device. The chart below indicates how to make such a reversal cable.

HK68 <u>Signal</u>	"D" pin <u>Number</u>	<u>Dir</u>	"D" pin <u>Number</u>	Modem <u>Signal</u>
Rcv Data	2	<-	3	Rcv Data
Tx Data	3	->	2	Tx Data
DCD	4	<-	6	DSR
DTR	5	->	20	DTR
RTS	6	->	4	RTS
Gnd	7		7	Gnd
CTS	20	<-	5	CTS
RI	22	<-	22	RI
-	-	<-	8	DCD

Table 27. Connection to a Data Set (Modem)

Note that three pairs of signals (pin 2-3, 4-6, 5-20) are reversed between the HK68 and the modem, although the signals at both ends have the same name. For some devices, DCD (pin 8) may need to be connected to RI on the HK68.

In order for the serial interface to work, the following conditions must prevail regardless of the device type which is connected. Otherwise, the port may appear to lock up. To avoid confusion, this listing is with respect to the HK68 end of the interface. A "TRUE" RS-232-C signal is a positive voltage.

HK68-SCC <u>Signal</u>	"D" pin <u>Number</u>	<u>Dir</u>	<u>Comments</u>
Rcv Data	2	<-	Data
Tx Data	3	->	Data
DCD	4	<-	Must be TRUE for HK68 to Receive (M10, V20)
DTR	5	->	Will be TRUE if port is open
RTS	6	->	Will be TRUE if port is open
Gnd	7		Signal Ground
CTS	20	<-	Must be TRUE for HK68 to Transmit (M10, V20)
RI	22	<-	Ring Indicator for on-card ports

Table 28. HK68 RS-232-C Interface

The HK68 Clear to Send signal (CTS) may be used to synchronize the HK68 with a slow device or one which may be remotely switched on- and off-line. The DCD and CTS status inputs to the HK68 (and SBX-SCC) have a default jumper, which can be set so that an unconnected input appears either "true" or "false". The normal default is "false", so that UNIX cannot dump characters out to an unconnected port.

By the way, if RS-232 "standards" confuse you, then you're normal.

16.4.2 CDC Eight Channel Expansion Board

The eight channel CDC expansion board has a slightly different interface. In particular, the DCD signal is used instead of RI for the modem interface.

The eight channel serial expansion boards allow either data terminal or data set devices to be connected without a special cable. This is achieved by jumper plugs on the expansion board which route the interface signals to the proper chips.

<u>8 Channel</u> <u>Board</u>	<u>Dir</u>	<u>"D" pin</u> <u>data term</u>	<u>"D" pin</u> <u>data set</u>	<u>Comments</u>
Rvc Data	<-	3	2	
Tx Data	->	2	3	
DCD	<-	8	8	Must be true (or open) to receive data
DTR	->	20	6	
RTS	->	4	5	
Gnd		7	7	
CTS	<-	5	4	Must be true (or jumpered) to transmit
DSR	<-	6	20	

Table 29. CDC Eight Channel Expansion RS-232-C Interface

Use the "data term" column when you are connecting the serial board to a data terminal. Use the "data set" column when you are connecting to a modem device. In either case, the ribbon cable does not need a reversal. There are a number of jumpers on the expansion board. Refer to the serial expansion board hardware manual for details. (Central Data, board model B1031).

External cables connect to Heurikon equipment according to the following table. (Note: refer to the particular hardware manual for details.)

Physical Device #	Name	Physical Device Loc (M10)	-----Connector Label-----		
			Minibox	MLZ-804/814	HSE
0	co	HK68 port B	B1	E1	CONSOLE
1	tty0	HK68 port A	C1	D1	S1
2	tty1	HK68 port D	B2	E2	S2
3	tty2	HK68 port C	C2	D2	S3
4	tty3	SBX-SCC P7 port B	B3	E3	S4
5	tty4	SBX-SCC P7 port A	C3	D3	S5
6	tty5	SBX-SCC P7 port D	B4	E4	S6
7	tty6	SBX-SCC P7 port C	C4	D4	S7
8	tty7	SBX-SCC P8 port B		E5	S8
9	tty8	SBX-SCC P8 port A		D5	S9
10	tty9	SBX-SCC P8 port D			S10
11	ttyA	SBX-SCC P8 port C			S11

Table 30. Serial Port Connections

16.5 Device File Setup

For M10 systems, the first four physical serial port interfaces are on the HK68 itself. The second group of four are assumed to be on an SBX-SCC serial expansion module connected to P7. The next four expansion ports, if used, are assumed to be on SBX P8. Note that in order to have more than eight ports, both SBX connectors must be used.

On V10 and V20 systems, only one (V20) or two (V10) serial ports are on-card. All others are on serial port expansion cards.

All on-card serial ports (except for one which is special) are major device zero. The desired serial port configuration is conveyed to the system via the minor device numbers, which are assigned to the device in the /dev directory.

The suggested port configuration data to use in creating the /dev entries for the serial ports on an M10 system is shown below. Of course, you may name the ports as you please; however, certain programs assume that the ports are named "tty...".

(format: mknod device type major minor)

mknod /dev/tty c 1 0 Active process tty

On-card Ports (M10):

mknod /dev/co c 0 0 Console device
 mknod /dev/lp c 0 1 Same as tty0
 mknod /dev/tty0 c 0 1
 mknod /dev/tty0m c 0 129
 mknod /dev/tty1 c 0 2
 mknod /dev/tty1a c 0 66
 mknod /dev/tty1b c 0 98
 mknod /dev/tty1m c 0 130
 mknod /dev/tty2 c 0 3
 mknod /dev/tty2a c 0 67
 mknod /dev/tty2b c 0 99
 mknod /dev/tty2m c 0 131

Expansion Ports: (SBX-SCC on P7)

mknod /dev/tty3 c 0 4
 mknod /dev/tty3a c 0 68
 mknod /dev/tty3b c 0 100
 mknod /dev/tty3m c 0 132
 mknod /dev/tty4 c 0 5
 mknod /dev/tty4a c 0 69
 mknod /dev/tty4b c 0 101
 mknod /dev/tty4m c 0 133
 mknod /dev/tty5 c 0 6
 mknod /dev/tty5m c 0 134
 mknod /dev/tty6 c 0 7
 mknod /dev/tty6m c 0 135

Expansion Ports: (SBX-SCC on P8)

mknod /dev/tty7 c 0 8

 mknod /dev/tty7m c 0 136
 mknod /dev/tty8 c 0 9

 mknod /dev/tty8m c 0 137
 mknod /dev/tty9 c 0 10
 mknod /dev/tty9m c 0 138
 mknod /dev/ttyA c 0 11

8 Chnl Expansion Ports: (CDC)

mknod /dev/tty10 c 12 0 board 1
 mknod /dev/tty11 c 12 1 ...etc...
 mknod /dev/tty20 c 12 8 board 2
 mknod /dev/tty23m c 12 139

Datasud Expansion Ports:

mknod /dev/ttyX c 8 0
 mknod /dev/ttyY c 8 1

Table 31. /dev/tty nodes (M10 Example)

The device "/dev/tty" may be used by a program as a pseudonym for the actual tty device in use. By using this designation, a program need not know which port is actually being used for the controlling terminal.

In general, the format for the minor device number is:

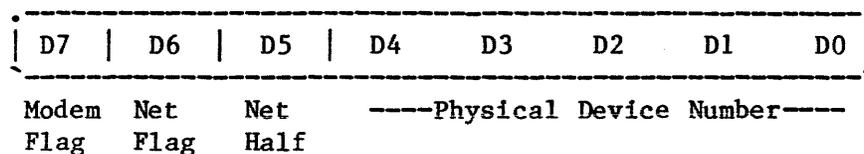


Figure 42. Serial Port Minor Device Format

For more information about minor device numbers, refer to section {17.3}. The minor device number for "mknod" is constructed by adding the desired bit values to create a decimal number.

There are two other serial port names used by "getty" and "init" to communicate with the system "console". They allow the console device to be assigned to whatever port or file you desire, instead of the /dev/console device (which is port B on the HK68.) They implement a "virtual" console. /dev/syscon is linked to /dev/systty according to an entry in /etc/inittab.

```

/dev/syscon
/dev/systty

```

Refer to section {10.10} for more information on those.

16.6 Sample System Configuration

This is an example of a typical system configuration, assuming there are three systems, with names "e1", "e2" and "sales". The system name is the "nodename", which may be set by the system administrator. (See section {10.8}.)

<u>System</u>	<u>Port</u>	<u>Devices</u>
"e1":	co	CRT-e1-Console
	tty0	Line Printer "e1"
	tty1	Network connection to/from system "e2"
	tty2m	Auto-answer modem
	tty2	modem dial-out
	tty3	User CRT-e1-3
	tty4	(not used)
"e2"	co	CRT-e2-Console
	tty0	Line Printer "e2"
	tty1	User CRT-e2-1
	tty2	Network connection to/from system "e1"
	tty3	Network connection to/from system "sales"
	tty4	(not used)
	tty5	User CRT-e2-5
"sales"	co	CRT-sales-Console
	tty0	Line Printer "sales"
	tty1	Network connection to/from system "e2"
	tty2	User CRT-sales-2

Table 32. Sample Network Configuration

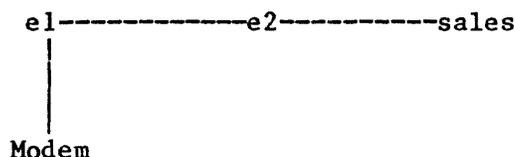


Figure 43. Sample Network Configuration

The three systems are linked together in a chain. System "e1" at one end, system "sales" at the other and system "e2" in the middle. Of course, the chain could be converted to a ring by adding a link between "e1" and "sales"; however, this is not necessary except to increase the overall system reliability in case system "e2" goes off-line. Adding another system connected to "e2" would create a "star" network.

To call system "e2" from "e1", you could use one of these commands:

- ⊕ `cu -ltty1b dir`
- ⊕ `uucp file e2\!file`

The `/etc/inittab` entries for these systems are shown below. Only the multiuser entries (init level 2) are shown. The baud rate arguments on

the /etc/getty commands refer to the lines in the /etc/gettydefs file, as explained in section {6.3}.

System "e1"

```
co::respawn:/etc/getty console 9600
t0::off:/etc/getty tty0 9600
t1::respawn:/etc/getty ttyla 4800
t2::respawn:/etc/getty -t 60 tty2m md_1200 ; modem
t3::respawn:/etc/getty tty3 co_9600 ;
t4::off:/etc/getty tty4 9600
t5::respawn:/etc/getty tty5 co_9600
t6::respawn:/etc/getty tty6 co_9600
```

System "e2"

```
co::respawn:/etc/getty console 9600
t0::off:/etc/getty tty0 9600
t1::respawn:/etc/getty tty1 co_9600
t2::respawn:/etc/getty tty2a 4800
t3::respawn:/etc/getty tty3a 4800
t4::off:/etc/getty tty4 9600
t5::respawn:/etc/getty tty5 co_9600
t6::off:/etc/getty tty6 co_9600
```

System "sales"

```
co::respawn:/etc/getty console 9600
t0::off:/etc/getty tty0 9600
t1::respawn:/etc/getty ttyla 4800
t2::respawn:/etc/getty tty2 co_9600
```

The "uucp" files (used by uucp, uux and cu) describing devices on system "e1" should look something like this:

File L-devices

```
DIR tty2 0 1200
DIR tty1b 0 4800
ACUVENTEL tty2 0 1200
```

File L.sys

```
e2 Any,2 tty1b 4800 tty1b "" @ login:-@-login: uucp
```

System "e2" uses port tty2b to call system "e1" and port tty3b to call the "sales" system. For more information about the uucp system, refer to the UNIX Administrator Guide and the uucp "hints" in section {16.8} below.

16.7 Changing Serial Baud Rates

This section explains the procedures for changing the default baud rates of the HK68 serial communication ports.

16.7.1 How to Change the Baud Rate Values Used by UNIX

Any ordinary baud rate value of 19,200 or lower may be used with UNIX. The actual character throughput is a function of the baud rate and the number of active processes. It is difficult to sustain baud rates above 9600 baud, although short bursts may be achieved. This is because of the amount of processing required by UNIX to handle each character. The best way to determine the maximum baud rate is to actually run the desired program and measure the character rate.

- [1] In single user mode, the console baud rate is initialized according to the value of the Hbug monitor configuration word (See Appendix A).
- [2] In multiuser mode, the console and other login "tty" ports are initialized to baud rate values as specified in `/etc/inittab` and `/etc/gettydefs`. For example, the `/etc/inittab` lines:

```
co::respawn:/etc/getty console co 9600
t3::respawn:/etc/getty tty3 co_4800
```

coupled with these `/etc/gettydefs` lines:

```
co_9600# B9600 # B9600 SANE TAB3 #\r\nlogin: #co_4800
co_4800# B4800 # B4800 SANE TAB3 #\r\nlogin: #co_2400
```

will cause the console to be initialized at 9600 baud and port `tty3` at 4800 baud. Although not shown above, a blank line is required between each line of the `gettydefs` file. These particular definitions will also cause `getty` to search for another rate, if it cannot make sense out of the login response. (The designation EXTA should be used for 19,200 baud (not "B19200".) For more details, refer to `stty(1)`, `init(1m)`, `inittab(4)`, `gettydefs(4)` and `getty(1M)` in the UNIX User's Manual and Administrator's Manual. If you modify the `/etc/gettydefs` file, execute `~/etc/getty -c` to check for errors.

NOTE: DO NOT ATTEMPT TO CHANGE THESE FILES UNLESS YOU ARE VERY FAMILIAR WITH THEIR FUNCTIONS AND THE OPERATION OF THE UNIX EDITORS. You could easily cause gross improper system operation if these files are changed in an incorrect manner.

- [3] During a terminal session, the baud rate of your port may be changed as follows:
 - A. Type `~/stty nnnn` (where "nnnn" is the desired common baud rate value)

- B. Change your terminal to the new baud rate.
- C. Hit the carriage return key a couple of times to get a new prompt.
- D. If you reset your terminal, it may be necessary to type `^tset^` to reinitialize your terminal characteristics, prior to using `"vi"`.

The original baud rate (as specified by `inittab`) will be re-established if you logout.

- [4] The baud rates of non-login ports (e.g., an output port) can be changed by using the `stty` command. It may be necessary to put the port to sleep in the background to prevent the baud rate from being automatically reset to 9600. Example:

```
sleep 10000 </dev/tty0 &  
stty 1200 </dev/tty0
```

A good place to put the above commands for automatic execution would be in `/etc/rc`. A larger sleep time should be used to make the baud rate change "permanent". Be sure there is no "getty" running for such a port, by adjusting the `/etc/inittab` file. There is an sample of such a line in the `/etc/inittab` example, shown earlier in this guide.

- [5] Printer baud rates should be set via a `stty` command in the appropriate file in the `/usr/spool/lp/interface` directory. See `lpadmin(1m)`, et al., and the UNIX Administrator Guide for more information on the line printer logic. See section {16.10} in this guide for installation hints on the LP spooler system.
- [6] The current baud rate and other parameters of any port may be checked by using the `stty` command, as follows:

```
stty -a </dev/tty2
```

16.8 The UUCP System - Hints

The objective of this section is to fill in some gaps in the UNIX documentation and give some advice on debugging the uucp system.

See also:

- ♣ "Uucp Administration" in the UNIX Administrator Guide. This is the best material on installing the uucp system.
- ♣ "Uucp System" in the UNIX Support Tools Guide.
- ♣ "Cu Usage" in section {16.9} of this guide.
- ♣ "UUCP" manual page in section one of the UNIX User's Manual.

These hints are in random order. They represent solutions or ideas which we had while installing uucp.

- [1] In addition to the standard distribution files, you may need to create (via "mkdir" or "touch") these directories and files. All should be owned by "uucp" and belong to group "uucp".

```

drwxrwxrwx /usr/spool/uucppublic
drwxrwxrwx /usr/spool/uucp
drwxrwxrwx /usr/spool/uucp/.XQTDIR
drwx----- /usr/lib/uucp
-rw-r--r-- /usr/lib/uucp/L_stat
-rw-r--r-- /usr/lib/uucp/L_sub
-rw-r--r-- /usr/lib/uucp/R_stat
-rw-r--r-- /usr/lib/uucp/R_sub

```

- [2] A "no shell" message could be because a script in /usr/lib/uucp does not have "group" and "other" read permissions. Also, try changing the mode of that directory to 755.
- [3] All files should be owned by uucp.
- [4] /etc/passwd must have read permissions for group and other. That is the standard UNIX configuration, since /etc/passwd has "public" information in it.
- [5] "ACCESS (DENIED)" messages mean you or uucp do not have permission to create or read files in the directory you specified. The best thing to do is to move files back and forth between machines by using the /usr/spool/uucppublic directories at both ends.
- [6] Some handy commands to have around (as aliases):

```
alias L grep "\!*" /usr/spool/uucp/LOGFILE
alias mL tail -f /usr/spool/uucp/LOGFILE
alias uutest /usr/lib/uucp/uucico -r1 -x4 &
alias rmS rm /usr/spool/uucp/STST.*
alias rmCD rm /usr/spool/uucp/[CD].*
alias spool cd /usr/spool/uucp
alias cd cd /usr/spool/uucp
```

- [7] If you ever get "Shere=..." you forgot the "-r1" option on a manual uucico command. You will have to wait a minute or two for uucico to get tired and timeout.
- [8] If you run a manual `uucico -r1` command for testing, put it in the background, so you can kill it (via `kill 0`), if it hangs. Uucico is the workhorse for the uucp system. It handles all the protocols used to dial out, login on the remote machine and transfer files, and it checks the integrity of the data transfer.
- [9] Try `uucico -r1 -x4 -ssystem` to initiate a manual call to a particular system. Remove any STST* files in the spool directory first.
- [10] If the time stamps on messages or log entries are incorrect, check the "TZ" variable in `/usr/lib/uucp/uushell`.
- [11] If uucico gives "RETRY TIME NOT REACHED" in the LOGFILE, then remove the STST.* files from `/usr/spool/uucp` to allow testing to continue. The STST.* files prevent a retry until at least 10 minutes have elapsed, regardless of the retry time specified in L.sys.
- [12] Check your USERFILE, L.sys and L-devices very carefully, and check them often.
- [13] "NO (DEVICE)", "tty open failed" or "tty open did not work" errors could be because the tty* device, which is used for calling out, is not owned by the uucp "user". Uucico leaves the tty device with mode 600, and, if it doesn't own it, uucico can't use it the next time. These errors could also be due to the baud rate specifications in L.sys and L-devices not being in agreement.
- [14] An error such as: "uucp XQT DENIED (rmail jeff)" means that the program "rmail" isn't listed in the L-cmd file. It may also be due to a problem with the USERFILE.
- [15] If you get something along these lines:

```
uucp!e2 ... ret (400) from e2!root (MAIL FAIL)
```

it probably means that uuxqt cannot create, find or write a temporary file someplace. Be sure the `/usr/spool/uucp/.XQTDIR` directory exists and has mode 777. Also, the mode of

/usr/spool/uucp must be at least 755.

- [16] If mail or uucp transfers will only go in one direction or if the "Remote from ..." line in mail files is wrong, it may be because your "nodename" is incorrect. Use /etc/chgnod and reboot to correct your nodename. Refer to "Checking the Nodename", in section {10.8}. Remember to do that if you copy a /unix file from one system to another. To check the current value of the nodename, enter `uname -n`.
- [17] If uucp works when a password is not required, but does not work if the uucp login has a password, then try this line for the appropriate entry in /etc/gettydefs:

```
md_1200# B1200 # B1200 CS8 CREAD IGNPAR ISTRIP
ICRNL IXON IXANY ISIG ICANON ECHO ECHOE ECHOK
OPOST ONLCR TAB3 #\r\n\nHeu V\r\nlogin: #md_1200
```

- [18] Have separate lognames for various uucp logins. For example, local machines could use "luucp" and external machines "euucp". This will allow you to change or delete logins for security reasons.

Remember, Man always wins in the end! (Although UUCP may stretch your endurance to the limit!)

16.8.1 Uucp Modem Support

The uucp logic will support a variety of autodial modems. The "device" field in the L.sys file is used to indicate the modem type (e.g., "ACUVENTEL" means a Ven-Tel modem); and, the "phone" field has the telephone number dialing code for the modem. At this time, these manufacturers are supported:

Bizcomp	Datec	Hayes
HayesDT	HayesDP	Novation
SmartCat	Vadic	Ven-Tel
USRobotics		

Use "ACUHAYES" or "ACUVENTEL", for example, in L.sys and L-devices to reference those devices.

Sometimes, a "smart" modem may get itself into a "zombie" state where it will not accept calls, yet it is not in use. To reset the modems at regular intervals, put a line like

```
56 * * * * echo </dev/tty2
```

in /usr/lib/crontab for each modem line. The "sleep" will cause the kernel to briefly drop DTR (which will reset the modem), and start another getty if the port is not in use. It has no effect if the modem is in use.

16.8.2 Uucp Control Files

Here are some example /usr/lib/uucp control files: (Note: some of the lines in the L.sys file have been split for clarity.)

```
e2 Any,5 tty2b 4800 tty2b "" @ login:-exit-login: uucp
cust Any,2 ACUVENTEL 1200 2771234% \
"" @ login:-@-login: uucp3 ssword: Nathan
ihnp4 Any0400-0800 ACUVENTEL 1200 13125552171%% \
in:-@-in: kuucp word: Blimp
```

Figure 44. /usr/lib/uucp/L.sys file

```
DIR      tty2      tty2      1200
DIR      tty1      tty1      1200
ACUVENTEL tty2      tty2      1200
ACUVENTEL tty1      tty1      1200
```

Figure 45. /usr/lib/uucp/L-devices file

```
luucp, /
uucp, /usr/spool/uucppublic
kuucp,ihnp4 /usr/spool/uucppublic
kuucp, /usr/spool/uucppublic
, /
```

Figure 46. /usr/lib/uucp/USERFILE file

16.9 ^Cu^ Usage

- ♣ Note: This procedure is presented here because it is typical of that used to initiate a manual call to another system. You should use the standard uucp commands instead of this procedure for most activities, such as file transfers.

Do not confuse the "%take" and "%put" routines built into the "cu" program with the "uucp" (UNIX to UNIX Copy) file transfer program. "Cu" allows a manual connection to be made for an interactive session on a remote machine; while "uucp" automatically performs processor to processor transfers. The "cu" "%take" and "%put" allow only ASCII file exchanges and perform no error checking, while Uucp will transfer binary and/or ASCII files. Summary:

- ♣ "Cu" is for manual, interactive connections to remote systems.
- ♣ "Uucp" is the best way to move data between two systems.

NOTE: This example assumes port "tty2" is connected to the modem. The actual modem port will be a function of your system configuration. E.g., you may be using "tty0" on a V10 system. Remember that a port name such as "tty2m" is used for incoming calls only. Devices "tty2" and "tty2m" refer to the same physical port.

- [1] (This step is not required if "cu" will be used throughout the session. That is, only do this if you need to disconnect "cu" and still maintain the telephone link.) Enter:

```
sleep 2000 </dev/tty2 &
```

This starts a "sleep" running in the background attached to the modem port to keep DTR on even if "cu" is terminated. The number of seconds should be chosen to be just greater than the expected session length. This will cause the modem to be released even if you forget to kill the sleep later.

- [2] `cu -ltty2 dir`. This will connect you to the modem. The getty running on tty2m, if any, will be killed by this command (or the "sleep", if used). A new getty will start, but it will be blocked by the serial device driver.

- ♣ The L-devices file must have a "DIR" line for any port used by ^cu^. Refer to figure {45}, above.

- [3]

Dial the desired number. Login on the remote system and conduct your session.

- ♣ The ^cu^ dial function and the dial(3) subroutine are not supported. You (or your program) must send the dialing

information to the modem.

- [4] To disconnect, enter `~.`. (That's a "tilde-dot carriage return".) This brings control back to the local system. If you did a "sleep" on the port earlier, the modem will maintain the connection. To reconnect to the modem, just do another `cu -lty2 dir`.

♣ Caution: if you are using `cu` from another `cu` or over Ethernet, be sure to add another `~` to your disconnect command (e.g., `~.`). Otherwise, you will disconnect only the closest `cu` or `rlogin`.

- [5] If you started a "sleep", above, kill the "sleep" process to release the modem and the telephone connection. If you've forgotten the sleep's process ID, enter "wait" followed by the delete key to get a listing of the background processes, or just enter `kill 0`. The "getty" for `tty2m`, if one was running, will restart and incoming calls will be recognized.

If you have trouble getting `cu` to work, check section {16.8}.

16.10 The LP Spooler Logic

There are numerous ways to connect a printer to the HK68 and program UNIX to output a file to the printer port.

Connection Methods:

- ♣ Connect to a serial port, such as `/dev/tty0`. This method allows a wide range of printers to be used. The only interface requirement is RS-232-C compatibility. Any port used for a printer should not have a "getty" running. That is, it should not be assigned as a login port by `/etc/inittab`.
- ♣ Connect to the Streamer/Centronics parallel port, `"/dev/cent"`. This is a Centronics-type interface, supported on some Heurikon systems.

On the M10, this connector is usually used for a streamer tape. It can be used for either a Streamer tape or a printer, but not both. A special adapter cable is needed to connect the HK68 to a Centronics-type printer. See the HK68 User's Manual for details and the CENT(7) manual page at the end of this guide.

On V10 or V20 systems, a separate board is required to support the Centronics interface.

- ♣ Connect the printer to the "auxiliary" port of your CRT terminal, if it has one. Then, write a short program, which will command your terminal to transfer any data received directly to the auxiliary port. (This can be done with the LP Spooler logic, described below, or via a standalone program under your own control.) This has the

advantage of not using up a separate port for each printer; however, you probably will not be able to use your terminal while the printing is in progress. This method works well in situations which need a small "personal" printer at each terminal for printing short files or mail.

UNIX Printing Facilities:

- ♣ The Line Printer Spooler logic, "lp". This is a package of routines which can handle one or more printers and of various types. It is described in more detail below.
- ♣ The "lpr" program, designed to work with one printer on port "/dev/lp". This is a simple system, which supports only one printer, and does not allow customization. Since it is inferior to the "lp" system, we are not providing details here. Consult the UNIX User's Manual for more information on lpr.
- ♣ You can provide your own programs, if you wish. The most trivial case would be to "cat" directly to the port, as in:

```
cat file >/dev/tty2
```

You must take care when using this method to set the baud rate and other port characteristics correctly. Refer to section {16.7} for information on setting the baud rate. You may need to use the "stty" command to enable output post processing and insertion of carriage returns wherever a line feed occurs, as in:

```
stty opost onlcr < /dev/tty2
```

- ♣ The "pr" program is helpful in formatting a file prior to using lp or lpr. It will add line numbers (-n) and headings, and can do special manipulations on the output file, such as expanding tabs (-e) and text positioning (-o).

16.10.1 Printing Via the LP Spooler

The UNIX LP Spooler facility allows numerous printers of various types to be used. Files may be printed on any printer, by any printer in the same "class" or only on a particular device.

This flexibility is achieved by having a separate interface program for each printer or class of printers. The system administrator can assign each printer a port, a class and an interface program.

There is a group of programs associated with the spooler logic. Some of them are described below along with installation instructions. This guide does not intend to answer all the questions about the LP Spooler system. The discussion below should only be used as an example. The complete story may be found in "LP" section of the UNIX Administrator Guide. Be sure to read it. Also, for system V.2, look at /etc/lp/README.lp.

Installation instructions - Example and Summary:

- [1] When shipped from Heurikon, your lp spooler may already be enabled. Before doing any configuration changes, you must turn off the scheduler via `~/usr/lib/lpshut`.

- [2] If you want to remove the default "tosh" (Toshiba) destination, execute:

```
~/usr/lib/lpadmin -d      (removes default destination)
~/usr/lib/lpadmin -xtosh (removes destination tosh)
```

- [3] Execute the "lpadmin" command according to the needs of your situation. For example, the following commands will install a TI-810 printer on port `/dev/tty3` and pull in the "dumb" interface model. The second command assigns the printer as the "default" so it does not have to be explicitly named with each "lp" command.

```
lpadmin -pti810 -mdumb -v/dev/tty3
lpadmin -dti810
```

When you execute the "lpadmin" command, you must be the super-user or have changed to "lp" via `~/su lp`.

- [4] Go to the interface directory and edit the printer program (script) as needed for your particular needs. In this example, the interface program will be `~/usr/spool/lp/interface/ti810`, a listing of which appears later in this section. The interface program actually does the work, including setting up the proper serial port attributes and baud rate. The interface scripts should be mode 755.

- [5] Execute "accept" and "enable" commands to turn the printer on.

♣ When using "accept", a message like "printer pr1 has disappeared!" indicates that the printer specified in the accept command was not recognized as being a valid destination; it is not in the qstatus file. Be sure you are typing the command properly. Try clearing the qstatus and pstatus files and repeating the `~/lpadmin` commands.

- [6] Become the super-user and start the scheduler ("lpsched") via the following commands:

```
rm -f /usr/spool/lp/SCHEDLOCK
~/usr/lib/lpsched
```

Those two commands can be put in `/etc/rc` to make them automatic when booting. They are probably already there.

- [7] If you do not want interrupted jobs to be restarted after a reboot, add this line to the beginning of the above command sequence:

```
rm -f /usr/spool/lp/request/**
```

- [8] Use `lpstat -t` to check the status of the lp system.

If you have trouble installing a printer, sometimes it helps to remove all files (except those in the model directory) and start over. Also, try running `vchk` to check your system configuration. These manual steps may be useful in debugging your system:

- [1] Check the `/usr/spool/lp` directory to be sure you have the following files and subdirectories. All files and directories should initially be empty, except the "model" directory. All files and directories should be owned by "lp".

<u>permissions</u>	<u>dir or file name</u>
drwxr-xr-x	class
drwxr-xr-x	interface
drwxr-xr-x	member
drwxr-xr-x	model
drwxr-xr-x	request
-rw-r--r--	outputq
-rw-r--r--	pstatus
-rw-r--r--	qstatus
-rw-r--r--	seqfile

Figure 47. `/usr/spool/lp` directory

- [2] Create those you do not have by using one of the following command sequences:

<code>mkdir dirname</code>		<code>touch filename</code>
<code>chmod 755 dirname</code>	or	<code>chmod 644 filename</code>
<code>chown lp dirname</code>		<code>chown lp filename</code>
<code>chgrp lp dirname</code>		<code>chgrp lp filename</code>

- [3] The following files should be set as indicated:

<u>file</u>	<u>mode</u>	<u>user</u>	<u>group</u>
<code>/usr/lib/lpadmin</code>	755	lp	bin
<code>/usr/lib/lpsched</code>	755	lp	bin
<code>/usr/bin/lp</code>	6755	lp	lp

- [4] The "lp" line in the `/etc/passwd` file should be:

```
lp:xxxxxxxxxxxx:7:2:lp:/usr/spool/lp:
```

⊕ Note: The previous two items are contrary to the UNIX documentation on lp. Believe what you will, but the proof is in the pudding.

- [5] Be sure there is a `sleep 2` at the end of `/etc/rc`. Otherwise, `lpsched` may not take hold.

After the LP system has been installed, the "lp" command can be used to start printing, as in these examples:

```
lp filename
lp -t"These are Titles" filename
pr -fn file1 file 2 | lp
```

Here is a typical interface program. There are others in the `/usr/spool/lp/model` directory.

```
# lp interface for TI 810
# ./ti810 lpid username title copies options files
stty 1200 tab3 opost onlcr 0<&l
x="XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
echo "\014\c"
echo "$x\n\n\n"
banner "$2"
echo "\n"
user=`grep "^$2:" /etc/passwd | line | cut -d: -f5`
if [ -n "$user" ] ; then
    echo "User: $user\n"
else
    echo "\n"
fi
echo "Request id: $1      Printer: `basename $0`\n"
date
echo "\n"
if [ -n "$3" ] ; then
    banner $3
fi
copies=$4
echo "\014\c"
shift; shift; shift; shift; shift
files="$*"
i=1
while [ $i -le $copies ]
do
    for file in $files ; do
        cat "$file" 2>&l
        echo "\014\c"
    done
    i=`expr $i + 1`
done
exit 0
```

Figure 48. Sample LP Interface Program

16.11 Stty Options

The `stty` command allows you to adjust certain aspects of your serial port interface. The `stty -a` command will print out the current state of the variables. Here is a list of some of the variables, and their "normal" states (a "-" means that flag is off):

```

speed  9600 baud
intr   DEL           (Pgm interrupt, stop)
quit   ^|           (Pgm interrupt, dump)
erase  ^h           (backspace)
kill   @            (line erase)
eof    ^d           (keyboard end-of-file)
eol    ^^

```

```

-parenb -parodd cs8 -cstopb -hupcl cread -clocal
-ignbrk brkint ignpar -parmrk -inpck istrip isig
-inlcr -igncr icrnl -iuclic ixon ixany -ixoff opost
icanon -xcase echo echoe echok -echonl -noflsh
-olcuc onlcr -ocrnl -onocr -oniret -ofill -ofdel

```

Table 33. `stty` options

Some of these flags control conversion of the carriage return key to a line feed, how the backspace character is handled (just backspace or backspace and delete on the CRT), the baud rate, parity checking and whether the Control-S scroll-stop logic (ixon) is enabled.

Refer to the "stty(1)" manual page in the UNIX User's Manual for a complete description of these flags. There are example "stty" commands in figure {48} and section {8.17}. Note that for an stty command to "stick", the port must be "open". This will be the case if the tty port is active (user logged in) or put to "sleep" in the background, as described in section {16.7}. To see if the values are sticking, run `stty -a </dev/device>`.

The "SANE" option for stty (used in `stty sane` or in `/etc/gettydefs`) is a composite of the following options:

```

brkint, ignpar, istrip, icrnl, ixon, opost, onlcr
cs7, parenb, cread, isig, icanon, echo, echok

```

To set the terminal options from a program, use the "ioctl" call, described in the `termio(7)` section of the UNIX User's Manual. This example sequence turns off canonical input processing, enables ixon, changes the baud rate and causes the immediate delivery of each received character (no buffering).

```

#include <sys/types.h>
#include <errno.h>
#include <termio.h> /* be sure to study termio(7) */
#include <fcntl.h>

int    fdtty;
struct termio tio;
#define TTY    "/dev/tty1" /* could use /dev/tty */

main()
{
    if ( (fdtty=open(TTY,O_RDWR)) < 0 ) {
        perror("yy: tty open fail");
        exit(1);
    }
#ifdef MODIFY
    /* this code will modify modes on a previously open port */
    if ( ioctl(fdtty,TCGETA,&tio) < 0 ) { /* get current modes */
        perror("yy: ioctl TCGETA fail");
        exit(1);
    }
    tio.c_lflag &= ~(ISIG|ICANON);
    tio.c_iflag |= IXON; /* enable IXON protocol */
    tio.c_cflag &= ~(CBAUD);
    tio.c_cflag |= B9600; /* change baud rate */
    tio.c_cc[VMIN] = 1; /* read each character ASAP */
#else
    /* this version will just set the modes as desired */
    tio.c_line = 0;
    tio.c_lflag = 0;
    tio.c_oflag = 0; /* no post processing */
    tio.c_iflag = IXON;
    tio.c_cflag = CREAD | CS8 | B9600;
    tio.c_cc[VMIN] = 1; /* read each character ASAP */
    tio.c_cc[VTIME] = 0;
#endif
    if ( ioctl(fdtty,TCSETA,&tio) < 0 ) { /* set the new modes */
        perror("yy: ioctl TCSETA fail");
        exit(1);
    }

    /* put real work here... */
}

```

Figure 49. Example Termio(7) Ioctl Calls

If you want to restore the tty settings to their original value, simply use TCGETA at the start of the program to save the original termio structure, then use TCSETA at the end to restore the values.

17. DEVICE NUMBERING AND NAMING CONVENTIONS

This section details the major and minor device numbers used in the Heurikon implementation of the UNIX Operating System.

♣ See also the "UNIX I/O System" in the UNIX Programming Guide.

17.1 Device Numbers and Types

UNIX device numbers consist of two eight-bit parts. These are called the "major" and the "minor" device numbers.

- ♣ The major device number references a class of devices, such as terminals or disk drives. Each device class is usually served by a separate device driver.
- ♣ The minor number specifies which device of the class is to be accessed, and indicates any special modes.

The device numbers correspond to a device name which is contained in the /dev file.

There are two types of I/O devices: character and blocked. Character devices (also known as "raw" devices) are the terminals and printers. Block devices are used for the Winchester disk and floppy drives. In general, character devices are slower than blocked devices although the raw (character) device interface is faster. (I.e., a character device is usually slow, physically. However, a raw interface to a block device is faster than the block interface to the same device.) Also, blocked devices are random access; character devices usually are not.

A blocked device also has a character device interface, which is called a "raw" device. A raw device is used for formatting and other special operations. See also section {11.1.7}.

17.2 Major Device Numbers

The following tables detail the major device numbers:

<u>M10</u>	<u>V10</u>	<u>V20</u>	<u>M220</u>	<u>Device</u>	<u>Manual Reference</u>
<u>Sys V</u>	<u>Sys V</u>	<u>V.2</u>	<u>V.2</u>		
0	0		0	SCSI (Winch, floppy, tape)	17.3.3
1				SBX Floppy (M10)	floppy(7)
2				Interphase 2190 SMD	it2190(7)
		0		Plessey-OMTI Winchester	wch(7)
	2	1		Interphase 3200 SMD	it3200(7)
			1	Ciprico Rimfire 2200 SMD	
		2		Plessey-OMTI Floppy	flp(7)
		3		Plessey-OMTI Streamer Tape	stp(7)

Table 34. Major Block Device Numbers

<u>M10</u>	<u>V10</u>	<u>V20</u>	<u>M220</u>	<u>Device</u>	<u>Manual Reference</u>
<u>Sys V</u>	<u>Sys V</u>	<u>V.2</u>	<u>V.2</u>		
0	0	0	0	Serial Ports	termio(7)
1	1	1	1	/dev/tty	tty(7)
2	2	2	2	mem, kmem, null	mem(7)
3	3	3	3	error log	err(7)
4	4		4	Raw SCSI Winchester, etal	17.3.3
		4		Plessey-OMTI Floppy	flp(7)
5				Raw SBX Floppy (M10)	floppy(7)
6				Centronics (P3)	cent(7)
7	7	5		68881 Floating Point	
			7	Ciprico Rimfire 2200 SMD	
	6	6		Datasud Centronics	cent(7)
	8	8		Datasud Dual Serial	
8				Streamer Tape (P3)	st(7)
9				CP/M Shell (Obsolete)	
10			10	Reel Tape (Ciprico)	mt-m10(7)
	10	10		Reel Tape (MCT)	mt-v(7)
11				User Jumpers and LEDs (M10)	dipled(7)
		11		Shell Layers, shl(1)	stx(7)
12	12	12	12	8 Chnl CDC Serial Expansion	
13	13	13		VRTX BusLink	
			6	MPC	mpc(7)
			13	MPC Link	link(7)
14				Interphase 2190 SMD	it2190(7)
	14	7		Interphase 3200 SMD	it3200(7)
15	15	9	9	Real Time Clock	rtc(7)
		14		Plessey-OMTI Winchester	wch(7)
		15		Plessey-OMTI Streamer	stp(7)
			14	MB II Interconnect	
20	20	20		Window, control tty, *wty	
21	21	21		Window, user tty, ttw*	
23	23	23		EXOS Ethernet xtty	xtty(4x)
24	24	24		EXOS Ethernet admin	admin(4x)
25	25	25		EXOS Ethernet xmem	xmem(4x)
26	26	26		EXOS Ethernet sockets	socket(4x)

Table 35. Major Character Device Numbers (RAW)

17.3 Minor Device Numbers

Minor device numbers are dependent on the particular I/O device driver. Some of the bits specify device characteristics, such as drive size; while others specify a physical unit number, such as drive 0 or drive 1.

The minor device value used with the 'mknod' command is the decimal equivalent of the binary pattern chosen from the following tables.

17.3.1 Serial Ports (HK68 and SBX-SCC only)

<u>Bit</u>	<u>Function</u>
7	Modem flag: 0 = std port, 1 = modem port
6	Network flag: 0 = std port, 1 = net port
5	Net Half flag: 0 = "a", 1 = "b"
4-0	Physical device number
	<u>Value</u> <u>Location</u>
	0-3 On-card
	4-7 SBX-SCC P7
	8-11 SBX-SCC P8
	12+ Not used

Table 36. Serial Port Device Assignments (on-card)

For details on the usage of the serial ports, refer to section {16}.

17.3.2 Floppy Disks - (SBX Module)

This section details the minor device number bit assignments for the SBX-Floppy Disk Controller module.

<u>Bit</u>	<u>Function</u>
7	Partition: 0 = normal, 1 = swap
6	High Density: 0 = off, 1 = on*
5	Sides: 0 = double, 1 = single
4	Base of root file system: 0 = cylinder 0, 1 = cylinder 2
3	Density: 0 = double or high, 1 = single
2	Size: 0 = 8 inch*, 1 = 5 inch
1,0	Physical unit number (MSB,LSB)

Table 37. SBX-FDIO Minor Device Assignments (M10)

Note: The high density option requires a factory hardware modification to the SBX-FDIO module. Also, 8" support is not available if the high density modification is done.

17.3.3 OMTI 5400 SCSI (M10, V10, M220)

This section details the bit assignments for the OMTI 5400 driver. This driver uses the "SCSI" interface.

<u>Bit</u>	<u>Function (or value)</u>
7,6,5	Controller number (0)
4	0
3	Physical Unit (0=w0, 1=w1)
2,1,0	Partition (000=a, ..., 111=h)

Table 38. OMTI 5400 - Winchester Minor Device Numbers (M10, V10, M220)

<u>Bit</u>	<u>Function (or value)</u>
7,6	Partition (00="b", 01="h", 10="a")
5	Size (0=5", 1=8")
4	1
3	0
2,1,0	Floppy Type (see below)

Table 39. OMTI 5400 - Floppy Minor Device Numbers (M10, V10, M220)

<u>Floppy Type</u>	<u>Sector Size</u>	<u>Sectors per Track</u> (<u>5-1/4"</u>) (<u>8"</u>)		<u>Density</u>	<u>Sides</u>
000	512	9	15	double	2
001	512	9	15	double	1
010	256	16	26	double	2
011	256	16	26	double	1
100	256	16	26	double*	2
101	256	16	26	double*	1
110	128	16	26	single	2
111	128	16	26	single	1

* single density on track zero

Table 40. OMTI 5400 - Floppy Type Values (M10, V10, M220)

<u>Bit</u>	<u>Function (or value)</u>
7,6,5	0,0,0
4	1
3	1
2,1	0,0
0	Rewind on Close (0=no, 1=yes)

Table 41. OMTI 5400 - Streamer Minor Device Numbers (M10, V10, M220)

Refer to section {12.10} for details on Winchester drive partitioning.

17.3.4 Streamer Tape

Streamer minor devices numbers are:

<u>M10</u>	<u>V20</u>	<u>Function</u>
0	2	Run forward to file mark on close
1	0	Rewind on close

Refer to st(7) and stp(7) in Appendix C for more details.

17.3.5 Reel-to-Reel Tape (Ciprico Tapemaster)

Some of the reel tape minor devices numbers are:

<u>Value</u>	<u>Function</u>
0	1600 BPI
1	1600 BPI, rewind on close
2	800 BPI Kennedy, 3200 BPI Cipher
3	800 BPI Kennedy, 3200 BPI Cipher rewind on close

Refer to the mt(7) pages in Appendix C for more details.

17.3.6 Ethernet

Refer to the separate documentation on the Ethernet interface for details.

17.4 Device Naming Conventions

The following list shows the correct device numbers to use for the "mknod" command. This list may not be complete, depending on your particular system configuration. Use it as a guide.

Sometimes, devices may be assigned more than one name either by having two separate nodes or by having two nodes linked together. The `ls -l` display can be used to determine the major and minor device numbers actually assigned to a particular device name.

For details on the serial ports, refer to section {16}.

(format: mknod device type major minor)

```
mknod /dev/tty      c 1 0
mknod /dev/console c 0 0
mknod /dev/lp       c 0 1 (for lpr)
mknod /dev/tty*    c 0 ? (varies)
mknod /dev/tty*    c 12 ? (varies)
mknod /dev/syscon  c 0 0
mknod /dev/systty  c 0 0
mknod /dev/wty*    c 20 ? (varies)
mknod /dev/ttyw*   c 21 ? (varies)
mknod /dev/ttyT*   c 23 ? (varies)
mknod /dev/ttyT8   c 23 8
```

(format: mknod device type major minor)

```

mknod /dev/mem      c 2  0
mknod /dev/kmem     c 2  1
mknod /dev/null     c 2  2
mknod /dev/rw0b     c 4  1 (M10)
mknod /dev/rw0c     c 4  2 (M10)
mknod /dev/rw0h     c 4 135 (M10)
mknod /dev/rw1h     c 4 143 (M10)
mknod /dev/rf5dd    c 5  4 (M10 SBX-FDIO)
mknod /dev/rf5sd    c 5 36 (M10 SBX-FDIO)
mknod /dev/rf5dh    c 5 68 (M10 SBX-FDIO)
mknod /dev/rf8dd    c 5  0 (M10 SBX-FDIO)
mknod /dev/rf8sd    c 5 32 (M10 SBX-FDIO)
mknod /dev/rf8ss    c 5 40 (M10 SBX-FDIO)
mknod /dev/rf5dd    c 4 16 (OMTI 5400)
mknod /dev/rf5sd    c 4 17 (OMTI 5400)
mknod /dev/cent     c 6  0
mknod /dev/ffp      c 7  0 (SKY)
mknod /dev/fpp      c 7  0 (68881)
mknod /dev/st0      c 8  0 (M10 P3)
mknod /dev/st1      c 8  1 (M10 P3)
mknod /dev/st0      c 4 24 (OMTI 5400)
mknod /dev/st1      c 4 25 (OMTI 5400)
mknod /dev/rtc      c 15 0 (M10, V10)
mknod /dev/rtciocl c 15 1
mknod /dev/mx0      c 9  0
mknod /dev/mt0      c 10 0
mknod /dev/mt1      c 10 1
mknod /dev/dipled   c 11 0

mknod /dev/swap     b 0  0
mknod /dev/w0b      b 0  1
mknod /dev/w0c      b 0  2
mknod /dev/w0h      b 0  7
mknod /dev/w1b      b 0  9
mknod /dev/w1c      b 0 10
mknod /dev/w1h      b 0 15

mknod /dev/f5dd     b 1  4 (M10 SBX-FDIO)
mknod /dev/f5sd     b 1 36 (M10 SBX-FDIO)
mknod /dev/f5dh     b 1 68 (M10 SBX-FDIO)
mknod /dev/f8dd     b 1  0 (M10 SBX-FDIO)
mknod /dev/f8sd     b 1 32 (M10 SBX-FDIO)
mknod /dev/f8ss     b 1 40 (M10 SBX-FDIO)
mknod /dev/f5dd     b 0 16 (OMTI 5400)
mknod /dev/f5sd     b 0 17 (OMTI 5400)

```

Table 42. mknod Device Summary - Partial

<u>Name</u>	<u>Meaning</u>	<u>Example</u>
co	console	co
tty	active tty	tty
tty*	terminals	tty3
lp	line printer	lp
cent	Centronics	cent
mem	memory	mem
kmem	kernel memory	kmem
null	null device	null
fsxy	Floppy s = size x = sides (s,d) y = density (s,d,h)	f5sd
swap	Swap Device	swap
wnx	Winchester	w0b
snx	SMD n = phy unit x = partition	s0b
rfsxy	Raw floppy s = size x = sides y = density	rf5dd
rwnx	Raw Winchester	rw0b
rsnx	Raw SMD n = phy unit x = partition	rs0b
fpp	68881 Floating Pt Proc	fpp
ffp	Sky Fast Floating Point Processor (M10)	ffp
tpx	Streamer Tape x=0 seek mark on close x=1 rewind on close	tp0
mxn	CP/M Shell Z80 Processor n = unit	mx0
mtn	R-R Tape n = function	mt0
rtc	Real Time Clock	rtc
dipled	Jumpers & LEDs	dipled
ttyTx	Ethernet x = port	ttyT0
wtyx	Windowing x = port	wty0
ttywx	Windowing x = port	ttyw0

Table 43. Device Naming Conventions

18. REFERENCE MATERIALS

Here are some books which we recommend you read for the basics and the details of UNIX. They are available at most computer and book stores (try: Uni-Ops Books, San Francisco, CA). Some are available from Heurikon.

- ♣ The Design of the UNIX Operating System, by Maurice J. Bach, Prentice Hall. This is an excellent book which covers UNIX internals. Required reading for those interested in how UNIX works. Covers UNIX architecture, scheduler algorithm, system calls, shared memory, context switching, memory management, device drivers and more.
- ♣ Using The UNIX System, by Richard Gauthier. Reston Publishing Company, Inc. In addition to explaining many fundamental concepts, this text includes some valuable information about system administration.
- ♣ The UNIX System, by S.R. Bourne. Addison-Wesley Publishing Company. Steve Bourne was one of the key individuals at Bell Laboratories responsible for the development of the UNIX Operating System. His book is a good, easy to follow, user's guide to UNIX.
- ♣ Introducing The UNIX Operating System, by McGilton and Morgan. McGraw-Hill Book Company. This book has some very good sections on using the text editor and document preparation features of the UNIX system.
- ♣ The UNIX Programming Environment, by Kernighan and Pike. Prentice-Hall. A good book for beginners and experienced UNIX users as well.
- ♣ UNIX Operating System Security, by F. T. Grampp and R. H. Morris, AT&T Bell Lab Technical Journal, Vol. 63, No. 4, Part 2, October 1984. This edition of the Technical Journal is devoted to UNIX topics.
- ♣ The C Programming Language, by Brian W. Kernighan and Dennis M. Ritchie. Prentice-Hall. This is a very well written and important book describing the syntax of the "C" language. This is the standard "C" language reference book.
- ♣ Exploring the UNIX System, by Stephen G. Kochan and Patrick H. Wood. Hayden Publishing. This is an excellent text based on system V. It has some sections on UNIX system security.
- ♣ Understanding UNIX, A Conceptual Guide, by James R. Groff and Paul N. Weinberg. Que Corporation. Easy reading for the neophyte to learn about the file system, administrative functions, etc.

19. APPENDIX A - Changing HK68 Serial Baud Rates

19.1 Background

The serial communication controller chips on the HK68 generate the baud rate frequency for asynchronous communications. Each port has its own internal baud rate generator. The generator divides the master clock input signal according to an integer value loaded by the software. The accuracy of the resulting baud rate clock is dependent on the master clock frequency and the particular value of the divider.

The default baud rate is set by the Hbug monitor program. The M10 and V10 monitor programs reads a pROM location for the default information. The V20 and M220 monitors read the NV RAM.

19.2 M10 and V10 Baud Rates

When the Hbug monitor (in pROM on the HK68) starts, it initializes the console CRT port (SCC port B) at 9600 or 19,200 baud. It computes the proper SCC divider value based on a configuration word in pROM, which tells Hbug what clock rates are being used on the HK68 and which baud rate is the default. The standard version is set for a 9600 console baud rate. The configuration word is also used by UNIX to initialize the other serial ports, and by UNIX whenever it is commanded to change port baud rates (via the stty command).

19.3 Changing the Hbug Configuration Word (M10, V10)

- [1] Remove the upper Hbug monitor pROM (HK68 ROM socket U27). Both bytes of the configuration word are contained in this ROM.
- [2] Load the contents of the pROM into a pROM programmer. Modify the appropriate locations according to the tables below.

<u>Byte offset</u> <u>in pROM L</u>	<u>Notes</u>
+0000 thru +0009	Do not change
+000A (hex)	High half of configuration word. Reserved. Do not change.
+000B (hex)	Low half of configuration word. Change as follows:

<u>MPU clock</u>	<u>SCC clock</u>	<u>Console default</u>	<u>ROM byte 000B</u>
10 Mhz	4.9152Mhz	9600 baud	15
10 Mhz	4.9152Mhz	19,200 baud	55
12 Mhz	4.9152Mhz	9600 baud	16
12 Mhz	4.9152Mhz	19,200 baud	56

Table 44. Hbug Configuration Word Values

Configuration word detail:

D15-D8 reserved (= 0)
 D7(MSB) = 0
 D6 = 0 for console default 9600 baud
 D6 = 1 for console default 19,200 baud
 D5,D4,D3,D2 = 0101 for 4.9152 Mhz SCC clock
 D1,D0 = 01 for 10 Mhz MPU clock
 D1,D0 = 10 for 12 Mhz MPU clock

Table 45. Hbug Configuration Word Detail

- [3] Program a new PROM and install it on the HK68.
- [4] The configuration word may be checked by using the Hbug "uc" command.

Although the configuration word is in PROM, it is copied to address 0x0004 in RAM when Hbug starts. The RAM value is used by the Hbug "uc" command and UNIX.

19.4 SBX-SCC Expansion Module Configuration

The SBX-SCC module(s) get their master clock from the HK68. The SBX-SCC module has a clock control jumper (J2 on the module), which should be set in the B position, J2-B. (This setting causes the module input clock to be divided by 2, which delivers 4.9152 Mhz to the SBX-SCC.) Note: The SCC chips on the SBX-SCC must be rated for 5 Mhz or more.

19.5 V20 and M220 Default Baud Rates

The V20 and M220 Hbug monitors read the default configuration information from the non-volatile RAM. To change the default values, the Hbug NV RAM commands may be used, according to the Hbug User's Manual.

20. APPENDIX B - Sed, Awk Usage Examples

This section details the `nroff` `mm` macro and shell script which were used to produce the index for this guide. The standard `mm` macros produce the Table of Contents, but there is no "built-in" facility for generating an index. Although these examples are for use with `nroff`, the primary purpose of this section is to show a real-life usage example of the `sed` and `awk` programs.

- ♣ The following "IN" macro definition was placed at the beginning of the `nroff` input file for the guide.

```
.de IN
.br
.nr nl -\\n(.v
.po 0
.ti 0
]\\$1 \\$2 \\$3 \\$4:\\nP
.br
.po
..
```

- ♣ The above macro was invoked throughout the document wherever an index key item occurred. For example, the lines:

```
.IN boot procedure, floppy
.IN floppy, booting
```

cause two items to appear in the index. The "IN" macro writes a special, left-adjusted line to the output file which contains the index key item and the current page number. These lines will be stripped out of the document before printing. The IN macro also sets the vertical line position back so the extra output line won't affect the page size for the "real" text.

- ♣ `Nroff` was run on the document source files using:

```
nice nroff -cm -rB2 -rW74 -e -T450-12 guide?.t >guide.f
```

See section {9.8} for other information about the `nroff` command.

- ♣ Within the formatted document, each index item starts with a right square bracket "]", which is used by the script listed on the following pages, to identify the index items and separate them from the formatted document itself. At this point, the index items are not sorted alphabetically; they are in page order and there may be some duplicates. Typical lines look like this:

```

]phys system call :91
]/dev/mem :91
]device drivers :91
]jumper.c program :92
]printf subroutine :92
]memory mapping :92
]/dev/kmem :93
    
```

◆ The script on the following pages takes the key items, sorts them, removes duplicates, combines page numbers for identical key items and creates a new text source file which, when fed back into nroff, produces the index. The script illustrates how a complicated function is easily implemented using existing UNIX tools. It took about one day to write, test and fine tune these programs. The script is installed in the documentation directory with names "do.guide", "print.guide", "do.index" and "print.index". They are linked together so that only one file needs to be maintained for the related functions.

The flowchart below illustrates the various relationships between the nroff files and the scripts.

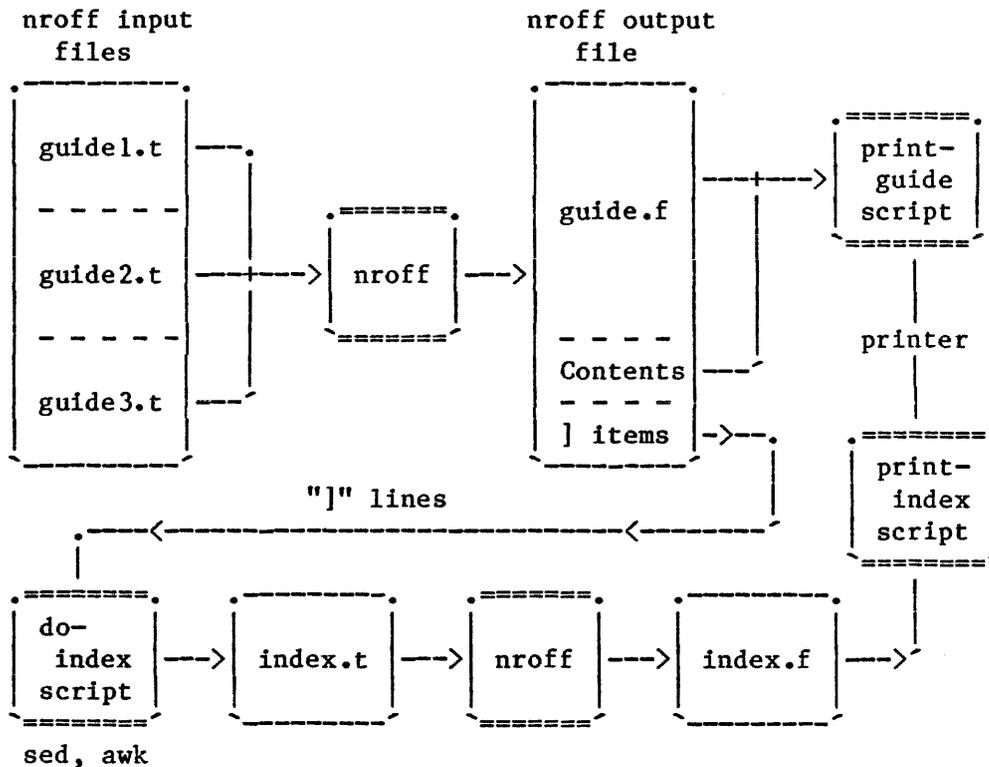


Figure 50. Guide and Index Preparation Flowchart

```

: # force /bin/sh
infile=guide.f
DOCDIR=/usr/local/jeff/doc
indext=$DOCDIR/index.t
indexf=$DOCDIR/index.f
awk=/tmp/awktemp

if [ $0 = "do.index" ] ; then

echo indexing
cat >$awk <<"AWK.SCRIPT"
  BEGIN {
    FS = ":"
    x=97 ; X=65 # "a" and "A", create upper case conversion array
    for (s = 0 ; s < 26 ; s++ ) { # for "a" thru "z"
      upper[sprintf("%c",x)]=sprintf("%c",X)
      upper[sprintf("%c",X)]=sprintf("%c",x)
      x++ ; X++
    }
  }
  {
    char = substr($1,1,1) # get first character of item
    if ( upper[char] == "" ) {
      char = substr($1,2,1) # use second char instead
      if ( upper[char] == "" ) {
        char = substr($1,3,1) # use third char instead
      }
    }
    CHAR = upper[char]
    if ( previous == $1 ) { # compare items, input stream is sorted
      temp = line
      line = sprintf("%s,%s", temp, $2) # add another pg number
    } else {
      if ( line != "" ) # don't do first time
        printf(".tl ^%s^\n", line) # output finished line
      line = sprintf("%s^%s", $1, $2) # starting new item
      if ( CHAR != pchar ) { # starting new letter
        printf(".P\n") # space, force page near bottom
        printf("%s\n",CHAR)
        printf(".br\n") # force letter out
      }
    }
    pchar = CHAR
    previous = $1
  }
  END { printf(".tl ^%s^\n", line) } # flush last item
AWK.SCRIPT

```

Figure 51. Nroff INDEX script - part 1

```

cat >$indext <<-"NROFF.HEADER"
    .SA 1
    .PH "I-\\nHeurikon UNIX - Reference Guide-I-\\n"
    .PF "Copyright 1987 Heurikon Corporation Madison, WI"
    .OH "INDEX"
    .EH "INDEX"
    .2C
    .lt 34
NROFF.HEADER

    sed -n -e "s/^\\](.*)/\\1/p" $infile | \
        sort -duft: +0 -1 +ln | awk -f $awk >> $indext
    nroff -cm -rW74 -e -T450-12 $indext > $indexf
    rm $awk
    exit

elif [ $0 = "print.index" ] ; then
    cat $indexf | col -x | under | \
        lp -t"UNIX Reference Guide INDEX"
    exit

elif [ $0 = "do.guide" ] ; then
    nroff -cm -rB2 -rW74 -e -T450-12 guide?.t > guide.f
    exit

elif [ $0 = "print.guide" ] ; then
    echo printing
    sed -n -e "/^\\]/!p" $infile | \
        col -x | under | lp -t"UNIX Reference Guide"
    exit

fi

```

Figure 52. Nroff INDEX script - part 2

The "col" and "under" programs are explained in section {9.8}.

21. APPENDIX C - Other Information21.1 Additional Documentation

The pages at the back of this guide detail certain additional commands and features of Heurikon UNIX. They do not appear in the permuted index.

badblk(1)	Bad Block utility (V20)
bootname(1)	Display unix file name (M10, V10)
clock(1)	Real Time Clock utility
fdref(1)	Floppy disk reformatter (M10)
mt(1)	
mtforeign(1)	Determine tape blocking factor
mtrewind(1)	Reel-to-reel tape rewind
mtskip(1)	Reel-to-reel tape skip
stape(1)	
sterase(1)	Streamer erase command
stretention(1)	Streamer retention command
strewind(1)	Streamer rewind command
ststats(1)	Streamer status command
cent(7)	Centronics printer port (M10)
dipled(7)	User jumpers and LEDs (M10)
floppy(7)	Floppy disk devices (M10)
flp(7)	Plessey - OMTI Floppy (V20)
it2190(7)	SMD Controller (M10)
it3200(7)	SMD Controller (V10, V20)
mt-m10(7)	Reel-to-reel tape device (M10)
mt-v(7)	Reel-to-reel tape device (V10, V20)
rtc(7)	Real Time Clock module
scsi(7)	OMTI-5400 devices (M10, V10)
st(7)	Streamer tape device (M10)
stp(7)	Plessey - OMTI Streamer (V20)
wch(7)	Plessey - OMTI Winchester (V20)

21.2 Unsupported Commands

Although these commands appear in the permuted index, and there may be pages for them in the UNIX manuals, they are either not supported or are incorrect. For example, take(1) is a special UniSoft command and is not supported. Ethernet commands (such as rlogin) are supported but are documented in separate Excellan manuals; some of these are marked with "*".

Section 1 Commands:

ftp*	hostid	hostname	netstat	put
rcp*	remsh*	rlogin*	ruptime*	rwho*
take	talk	telnet	tp	updater
take7	put7	rtake	se	sar
vmail	xmail	adb(5.2)	captoinfo	coffbin
untic				

Section 2 System Calls

accept*	bind	connect*	gethostid	gethostname
getpeername	getsockname	getsockopt	listen	recv
recvfrom	recvmsg	select*	send*	sendmsg
sendto	sethostid	sethostname	shutdown	socket*

Section 3 Subroutines

endhostent	endnetent	endprotoent	endservent
getdtablesize	gethostbyaddr	gethostbyname	gethostent
getnetbyaddr	getnetbyname	getnetent	getprotobyname
getprotobyname	getprotoent	getservent	getservbyname
getservbyport	inet_addr	inet_lnaof	inet_makeaddr
inet_netof	inet_network	inet_ntoa	insque
killpg	rcmd	readv	remque
rexec	rresvport	ruserok	sethostent
setnetent	setprotoent	setservent	writv
dial			

Section 4 File Formats: hosts protocols services

Section 5 Miscellaneous Facilities

intro(5n)	arp	inet	ip	tcp	udp
-----------	-----	------	----	-----	-----

Section 7 Special Files: sxt

Section 8 Procedures

ftpd	ifconfig	netmail	netmailer	remshd*
rexecd	rlogind	route	routed	rwhod*
telnetd	tftpd	trtp		

Volume 8 Administrator Manual

Disregard the entire section in the Administrators Guide on Take/Put.

21.3 System Configuration Summary

<u>Feature</u>	<u>M10</u>	<u>V10</u>	<u>V20</u>	<u>M220</u>
MPU	68010/020	68010/020	68020	68020
MMU	68451	68451	68851	68851
UNIX	V.0	V.0	V.2	V.2
Winchester on-card adapter controller	NCR5380 OMTI-5400 Adaptec	NCR5380 OMTI-5400 Adaptec	- Plessey OMTI-5400	WD33C93 OMTI-5400
Floppy optional size	SBX-FDIO OMTI-5400 5", 8"	OMTI-5400 5"	OMTI-5400 5"	OMTI-5400 5"
Serial on-card expansion	SCC (4 ports) SBX-SCC (4) CDC MB1031 (8)	SCC (2 ports) CD23/3608 (8) Datasud (2)	MFP (1 port) CD23/3608 (8) Datasud (2)	SCC (2) SBX-SCC (4) CDC (8)
Streamer I/F optional	P3 OMTI-5400	OMTI-5400	OMTI-5400	OMTI-5400
SMD	IT2190 (Interphase)	IT3200	IT3200	Ciprico 2200 Rimfire
Reel-to-Reel	Ciprico Tapemaster A	MCT 6020	MCT 6020	Ciprico 2000 Tapemaster
Floating Pt	SBX-FPP	68881	68881	68881
Centronics	P3	Datasud	Datasud	T.B.D.
VRTX	ME	VE/V20/V2F	VE/V20/V2F	M2F
Ethernet	EXOS 101/201	EXOS 202	EXOS 202	

Table 46. System Configuration Summary

22. READER COMMENT FORM

We would appreciate any comments you have concerning this guide. Please let us know if you have found any errors or feel that certain sections should be expanded. Thank you.

Name: _____ Title: _____	
Company: _____ Date: _____	
Address: _____	
City: _____ State: _____ ZIP: _____	
Telephone: () - _____ - _____	
Section	Comments
Mail to: Heurikon Corporation 3201 Latham Drive Madison, WI 53713	

Ref Guide - Rev D

BADBLK(1)

UNIX 5.2

BADBLK(1)

NAME

`badblk` - check a disk for bad blocks (V20 Plessey version)

SYNOPSIS

`badblk` [`-fav`] `special` [`blockno ...`]

DESCRIPTION

`Badblk` checks a disk for bad tracks and if requested maps the offending tracks to alternates. It will try to save as much of the damaged tracks as possible, writing the salvaged information back to the assigned alternate. It will notify the user of blocks that it couldn't recover. It is therefore possible to run `badblk` on a disk which has a file system on it without reformatting. The block numbers of the unrecoverable blocks can be used to determine of which files they are a part (using `ncheck(1M)`). Any such files may not be recoverable intact.

The normal operation is to scan the whole disk represented by `special` for bad tracks (except for the alternate track region). If block numbers are given at the end of the command, `badblk` will only test the tracks associated with those blocks for defects. In any event, for each bad block, `badblk` will ask whether it should be mapped out.

The `f` option forces the given blocks to be treated as bad even if there are no defects on the associated tracks. This can be used to map out intermitently failing blocks.

The `a` option directs `badblk` to scan all of the disk including the alternate track area. This should probably be done immediatly after a disk format to check for possible bad alternate tracks.

The `v` option causes a verbose trace of operation to be given.

This routine is for a system with an OMTI-5400 and a Plessey host adapter.

SEE ALSO

`ncheck(1M)`, `badblksmc(1)`, `wch(7)`, `ioctl(2)`.

BUGS

There is currently no convenient way to avoid using bad alternates.

NAME

`bootname` - Display the name of the executing kernel

SYNOPSIS

`bootname` [`arg`]

DESCRIPTION

With no argument, `bootname` prints the name of the file which was loaded by the standalone bootstrap. The name will be anchored at root. If any argument is provided, `bootname` prints the complete string used by the loader, which includes the drive number and root block number.

EXAMPLES

```
ps -n `bootname` -ef
```

will allow `ps` to work properly, regardless of which kernel was booted.

Normal usage would be to put the lines:

```
set kernel=`bootname`
alias ps ps -n $kernel
alias pstat pstat -n $kernel
```

in your `.cshrc` file, or simply add

```
cp `bootname` /unix
```

to `/etc/rc`; both methods minimize the number of times `bootname` needs to be executed. The last usage example allows those UNIX commands which assume the operating kernel is `"/unix"`, such as `ipcrs(1)` and `lav(1)`, to operate properly.

DIAGNOSTICS

If the file name cannot be found or if it makes no sense, an error message will be printed on `stderr`, the default UNIX file name, `"/unix"`, will be output on `stdout` and the exit status will be non-zero.

CAVEATS

The long word at absolute memory address `0x000` contains a pointer which must not be destroyed in order for this command to work.

`Bootname` is operational as of Heurikon V.0 UNIX release 7a.

FILES

`/dev/kmem` to read memory

SEE ALSO

`ps(1)`, `nlist(3)`

AUTHOR

Heurikon Corporation

CLOCK(1)

UNIX 5.0/5.2

CLOCK(1)

NAME

clock - Real Time Clock (RTC) module utility

SYNOPSIS

```

clock
clock -u
clock -r
clock -D
clock mmddhhmm

```

DESCRIPTION

clock will read and display the current time and date as known to UNIX and the DS1216 RTC module. (Special hardware is required).

clock -u will read the current time as known to the RTC module and use that value to set the UNIX time and date. This command is restricted to the super-user.

clock -r will read the current time as known to UNIX and use that value to set the DS1216 Clock/Calendar Module. This command is restricted to the super-user.

clock -D starts the clock daemon which will try to keep the UNIX time in sync with the RTC module. The daemon will not run if the times disagree by more than five minutes. The daemon may be run continuously (it sleeps most of the time); only one daemon should be allowed to run at a time. UNIX time convergence rate is about one second per minute. Sending signal SIGUSR1 to the daemon (via `kill -16 pid`) will display some statistics. This command is restricted to the super-user.

clock mmddhhmm will set both the UNIX date and time and the RTC module. The argument format is the same as `date(1)`.

EXIT CODES

clock in its first three forms returns 0 if the RTC and UNIX agree to within one second as to the current time, otherwise 1 is returned. A system or other error returns 2. clock -D returns 0 if the daemon starts successfully, non-zero otherwise.

EXAMPLES

To set the RTC, first use the UNIX `date` command. Then, execute clock -r. If there is a power failure, execute clock -u after rebooting to reset the UNIX date and time.

The clock -u and clock -D commands may be put in `/etc/rc`.

FILES

```

/dev/rtc
/dev/rtdciotl

```

SEE ALSO

`date(1)`, `rtc(7)`

NAME

`fdref` - Unified Floppy diskette formatter (M10 - SBX-FDIO)

SYNOPSIS

```
fdref device [-s 128|256|512|1024] [-x mode]
           [-d distance] [-k skewfactor] [-t lasttrk] [-q]
```

DESCRIPTION

`fdref` formats 5-1/4 inch and 8 inch floppy diskettes on an HK68/M10 using a SBX-FDIO module. (Not for OMTI-5400 floppy). The character 0xE5 is written in the data fields of each sector.

Device must be a raw floppy character device.

Options (may be specified in any order):

`-s n` sets the sector size to n bytes. The default value is 512.

`-k skewfactor`

Set the skew factor to skewfactor. This parameter is used to compute the logical sector values for each physical sector. A skew factor of 1 (default) will generate logical sectors in the sequence 1, 2, 3, 4, etc. A skew factor of 3 will generate the sequence 1, 4, 7, 11, etc. See also the `-d` option, below. Use the `-x` option to display a complete list of sector maps.

`-d distance`

Set the distance between logical sectors to distance sectors. This parameter operates like the skew factor except it specifies how many sectors separate logical sectors. A distance of 1 will generate logical sectors in the sequence 1, 2, 3, 4, etc. A distance of 2 will generate the sequence 1, a, 2, b, 3, c, 4, etc., where a, b and c are other sector numbers (whose values depend on the total sectors per track). Use the `-x` option to display a complete list of sector maps. This parameter has priority over `-k` and is generally more useful than `-k` when trying to optimize diskette performance.

`-t lasttrk`

Stop formatting after track lasttrk is reached. The default value for an 8 inch drive is 76. The defaults for 5-1/4 inch drives are 79 for 96 TPI drives and 39 for 48 TPI drives. Tracks are numbered from 0 to the value specified.

`-x mode`

Set debug mode to mode. A value of 1 will print a complete map of logical sectors for all legal distance values. No formatting will be done. Other values of mode are illegal.

`-q`

Set quiet mode. The default is to generate some terminal activity to keep you aware of what's happening. This option will allow only error messages to be printed.

FDREF(1)

UNIX 5.0

FDREF(1)

EXAMPLES

```
fdref /dev/rf8sd -d 7
```

will format an eight inch, single sided, double density, 512 bytes/sector diskette using a distance of 7 sectors between logical sectors.

```
fdref /dev/rf8sd -s 1024 -x 1
```

will print a sector map for a floppy with 1024 bytes/sector.

The following sh script can be used to find the optimal skew or distance values for use with dd:

```
for dist in 1 2 3 4 5 6 7 8 9 10 11 12 13 14 ; do
    echo $dist
    fdref /dev/rf8dd -d $dist -t 10
    time dd if=/dev/f8sd of=/dev/null count=150
done
```

We have found that a distance value of 7 is optimal for 8 inch floppies (512 bytes/sector, double density) and for high density 5-1/4 inch floppies. A distance of 4 works well for double density 5-1/4 inch floppies (512 bytes/sector).

FILES

/dev/rfd* raw floppy devices

SEE ALSO

dd(1), floppy(7)

DIAGNOSTICS

Generally self-explanatory. The logical sector map for the specified skew factor or distance will be printed for skews other than 1. Fdref will complain about a device or data file which does not exist or has improper modes. The current track number will be printed to indicate activity unless turned off by the -q option.

AUTHOR

Heurikon Corporation

NAME

mtforeign, mtrewind, mtskip

SYNOPSIS

mtforeign

mtrewind

mtskip device

mtskip n device

mtskip -n device

DESCRIPTION

Mtforeign prints the size (in decimal bytes) of the next block on the tape. This information can be used with tar(1) -b option.

Mtrewind moves the tape head position to a point immediately prior to the first set of data on a tape. This point is known as the BOT or beginning-of-tape position.

Mtskip allows users of systems equipped with reel-to-reel magnetic tape drives to quickly move about on tapes containing multiple file marks. Such tapes are created by using the tape special file which does not rewind on close (mt0). The first form will move the tape head position forward to a point immediately following the next file mark. The second form will move the tape head position forward to a point immediately following the ``nth`` file mark. The third form will move the tape head position backwards to a point immediately preceeding the ``nth`` file mark encountered.

CAVEATS

When using the third form be aware that file marks are detectable in both the forward and backward directions. See the examples.

EXAMPLES

Given the tape created with the following commands:

```
mtrewind /dev/mt0
tar cvfb /dev/mt0 20 /usr/games
tar cvfb /dev/mt0 20 /usr/man
mtrewind /dev/mt0
```

The second tar set may be read directly by saying:

```
mtskip /dev/mt0; tar tvf /dev/mt0
```

or

```
mtskip 1 /dev/mt0; tar tvf /dev/mt0
```

The same tar set may be read a second time by saying

MT(1)

UNIX 5.0/5.2

MT(1)

```
mtskip -l /dev/mt0; mtskip /dev/mt0
```

That is, move the tape head position backwards through the data of the second tar set to a point immediately preceding the tape mark introducing the second tar set. The second mtskip advances the tape head position over this tape mark leaving the drive ready to read the second tar set.

FILES

/dev/mt0 - the character special file for non-rewinding magtape.

SEE ALSO

cpio(1), tar(1), mt(7)

DIAGNOSTICS

If the tape drive is offline or the driver has not been installed properly, the program will report as such. Other diagnostics are taken from the standard perror() list.

NAME

sterase, strewind, stretension, ststats

SYNOPSIS

sterase

strewind

stretension

ststats

DESCRIPTION

Sterase causes the write-enabled tape in drive 0 to become initialized to an erased condition. All information on the tape is lost. The QIC-02 standard for quarter-inch tapes specifies that all writing must be done on erased tape. The tape drives used in this product do not require that this condition be met. However, erasing used tapes before re-use is recommended. Control returns to the user immediately after the erase command is sent to the drive.

Strewind causes the tape to rewind to the ``beginning-of-tape`` position. Control returns to the user immediately after the rewind command is sent to the drive.

Stretension causes the tape to be repositioned to bot, then advanced to eot (end-of-tape) and then back to bot. This operation serves to place the tape under constant tension throughout its length thereby increasing reliability. Often, tape read errors will disappear after executing a stretension. This operation should be executed prior to the use of any used tape. Control is returned to the user immediately after the command is sent to the drive.

Ststats prints lots of interesting facts about the operation of the tape drive. This command is only available on M10 systems.

DIAGNOSTICS

Error messages will be directed to the system console.

SEE ALSO

st(7)

CENT(7)

UNIX 5.0

CENT(7)

NAME

cent - Centronics printer driver (M10 only)

DESCRIPTION

Cent is the Centronics printer driver. It controls a printer connected to the P3 connector on the HK-68/M10 processor.

The major device number is 6. Cent ignores the minor device number.

Two ioctl system calls apply. They have the form:

```
ioctl (fildes, command, arg)
int arg;
```

The argument arg is ignored by cent. The commands using this form are:

UIOCEXTE Turn extended errors on. Forces cent to print the cause of I/O errors on the console.

UIOCNEXTE Turn extended errors off. This is the initial state of the driver.

FILES

/dev/cent

SEE ALSO

lpd(1), exterr(1), ioctl(2), lp(1).

AUTHOR

Heurikon Corporation

NAME

dipled - user jumpers and LED interface (M10 only)

DESCRIPTION

The state of the HK68/M10 user jumpers may be read and the user LEDs may be set using this device. This device is only applicable to M10 systems.

For read(2), the returned character represents the state of the user jumpers. Jumper number "1" (when numbering 1 through 8) is in bit position D7. A "1" bit means the corresponding jumper is installed.

For write(2), the last character written will affect the LEDs as follows:

If the minor device number is 0, the lower four bits of the character will be used to set the four LEDs. LED number "1" is controlled by bit position D3. A "1" bit will turn the corresponding LED on.

If the minor device number is 1, 2, 3 or 4, only one LED will be affected. Bit position D0 will be used in all cases to set or clear the addressed LED.

An open(2) of this port will turn off any kernel usage of the jumpers and LEDs. The debug usage may be restored by executing an ioctl(2) with cmd equal to UIOEXTE.

SEE ALSO

Heurikon UNIX - Reference Guide
open(2), read(2), write(2), ioctl(2)

FILES

/dev/dipled	read all jumpers or set all LEDs
/dev/led1	set LED1
/dev/led2	set LED2
/dev/led3	set LED3
/dev/led4	set LED4

FLOPPY(7)

UNIX 5.0

FLOPPY(7)

NAME

floppy - Floppy disk driver (M10)

DESCRIPTION

Floppy is the SBX floppy disk driver for a HK68/M10 system. Up to four floppy disk drives may be connected to the system. The floppy driver's major numbers are 5 for the raw device and 1 for the block device.

The following table describes the available formats. All of these formats may be used on single or double sided diskettes.

<u>Drive</u>	<u>Density</u>	<u>Bytes/Sector</u>	<u>Sectors/Track</u>
5 1/4"	double	128	24
		256	16
		512	9
		1024	5
5 1/4"	single	128	16
		256	9
		512	5
		1024	2
5-1/4"	high	128	52
		256	26
		512	15
		1024	8
8"	double	128	52
		256	26
		512	15
		1024	8
8"	single	128	26
		256	15
		512	8
		1024	4

The minor device number is formed from the following table:

<u>Bit</u>	<u>Function</u>
7	Partition (0="b", 1="a" swap)
6	Density (0 = single or double, 1 = high)
5	Number of sides (1 = single sided)
4	Skip the first two tracks (1 = yes)
3	Density (0 = double or high, 1 = single)
2	8" or 5 1/4 " drive (1 = 5 1/4 ")
0, 1	Physical unit number

Note that the driver adjusts itself for sector size. It reads an ID field off the disk and uses the sector length information stored there.

FILES

/dev/f*, /dev/rf*

SEE ALSO

exterr(1), fdref(1), tar(1), cpio(1), mkfs(1)

NAME

/dev/flp[01][a-d] - floppy disks (V20)

DESCRIPTION

This is the driver for the floppy aspect of the OMTI 5400 disk controller when used with a Plessey host adapter. The driver supports one 5-1/4 inch drive.

The minor number is composed of three parts and can be represented like `^00tttpp^`, where `^tttt^` is the type designator and the `^pp^` is the partition designator.

PARTITIONS

Each drive has four partitions which are layed out as follows (units in blocks):

<u>Partition</u>	<u>(pp)</u>	<u>Start</u>	<u>Length</u>	
a	(00)	0	1360	(root)
b	(01)	0	1440	(whole drive)
c	(10)	1360	80	(swap)
d	(11)			(unused)

TYPES

The following table describes the mapping of disk type to characteristics:

<u>Type(tttt)</u>	<u>Sides</u>	<u>Density</u>	<u>bytes/sector</u>	<u>sect/trk</u>
0	1	Single	128	16
1	2	Single	128	16
2	1	Single	128 (trk 0)	16
	1	Double	256 (others)	16
3	2	Single	128 (trk 0)	16
	2	Double	256 (others)	16
4	1	Double	256	16
5	2	Double	256	16
6	1	Double	512	8
7	1	Double	512	9
8	2	Double	512	8
9	2	Double	512	9
10	1	Double	1024	4
11	2	Double	1024	4
12-15		not used		

IOCTLS

The driver understands one ioctl:

UIOCFORMAT takes a diskformat structure as argument, formats the disk. It formats the whole drive.

SEE ALSO

Heurikon UNIX - Reference Guide

IT2190(7)

UNIX 5.0

IT2190(7)

NAME

it2190 - SMD disk driver (M10)

DESCRIPTION

It2190 is a SMD disk controller driver for Multibus I. It connects up to four SMD disk drives using an Interphase 2190 controller.

The driver's major numbers are 14 for the raw device and 2 for the block device.

The minor device number selects drive type, physical unit, and logical unit as follows:

<u>Bits</u>	<u>Function</u>
3	Physical unit number
0-2	Logical unit number

Drive partitioning is as described in the Heurikon UNIX Reference Guide.

FILES

/dev/f*, /dev/rf*

SEE ALSO

exterr(1), fdref(1), tar(1), badblk(1), mkfs(1)

NAME

it3200 - SMD disk driver (V10, V20)

DESCRIPTION

It3200 is a SMD disk controller driver for the VME bus. It connects up to two SMD disk drives using an Interphase 3200 controller. The driver's major numbers are 14 for the raw device and 2 (or 1) for the System V (or V.2) block device. The minor device number selects physical unit and logical unit as follows:

<u>Bits</u>	<u>Function</u>
3	Physical unit number
0-2	Logical unit number

The physical unit number selects one of two disks drives connected to the system. The logical unit number selects the parttion. Partion sizes are described in the Heurikon UNIX Reference Guide.

EXAMPLES

This sequence of command will format the disk, test for bad blocks, and build a file system using the entire disk.

```
diskformat /dev/rit0e
badblk /dev/rit0eo (for V10)
badblksmid /dev/rit0e (for V20)
mkfs /dev/it0e (maxbn/2) 2 2
```

Examine the rebuild script for command details and examples.

FILES

/dev/it*, /dev/rit*

SEE ALSO

badblk(1), diskformat(1), mkfs(1), badblksmid(1)

Several ioctl system calls apply. They have the form:

```
int arg;
ioctl (fildes, command, arg);
```

UIOCEXTE

Turn extended errors on. Forces mt to print the cause of I/O errors on the console. This is the initial state of the driver. This call ignores arg.

UIOCNEXTE

Turn extended errors off. This call ignores arg.

HMT_REWIND

Issues a rewind command to the tape transport. This call ignores arg.

HMT_OFFLINE

Issues an off-line command to the tape transport. This call ignores arg.

HMT_WFM

Writes a filemark at the current tape position. This call ignores arg.

HMT_SKIP

Skips forwards or backwards a number of filemarks. Arg is a signed integer indicating the number of filemarks to be skipped.

HMT_SPACE

Spaces forwards or backwards a number of blocks. Arg is a signed integer indicating the number of blocks to be skipped.

HMT_FOREIGN

Returns the number of bytes in the next tape block. The tape position is advanced one block. The number of bytes is returned in arg.

HMT_REPORT

Returns the drive status and the Tapemaster error code for the last tape error. Arg is a structure defined in magtape.h having the following form:

```
struct mt_errors arg;
```

SEE ALSO

mtskip(1), mtrewind(1), exterr(1)

DIAGNOSTICS

If a Cipher drive is asked to skip backwards up to or past the BOT the mtskip command will produce a diagnostic. This message can be ignored.

MT-M10(7)

UNIX 5.0

MT-M10(7)

Diagnostics will be printed in the form of:

```
tm: ecode: nn drvsts: nn
```

Ecode will be a one or two digit hexadecimal number whose meaning can be found in the Heurikon Corporation Unix Reference Guide. Drvsts, which stands for drive status, will be a two digit hexadecimal number whose meaning can similarly be found in the Unix Reference Guide. The ``tm`` stands for TAPEMASTER.

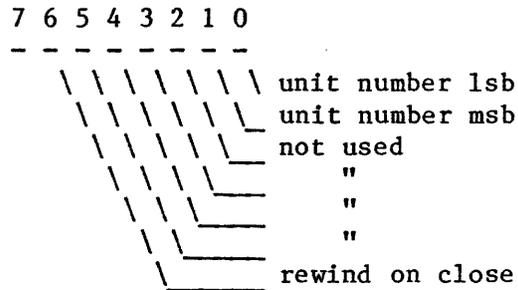
NAME

mt - a description of the magtape interface (V10, V20)

DESCRIPTION

Mt is the nine-track tape driver. It controls up to four tape transports connected to a MCT-6020 VME card.

The meaning of each bit of the minor device number is:



Several `ioctl` system calls apply. They have the form:

```
int arg;
ioctl (fildes, command, arg);
```

HMT_REWIND

Issues a rewind command to the tape transport. Arg is ignored.

HMT_WFM

Writes a filemark at the current tape position. Arg is ignored.

HMT_SKIP

Skips forwards or backwards a number of filemarks. Arg is a signed integer indicating the number of filemarks to be skipped.

HMT_SPACE

Spaces forwards or backwards a number of blocks. Arg is a signed integer indicating the number of blocks to be skipped.

HMT_FOREIGN

The tape position is advanced one block. The number of bytes in that block is returned in arg.

SEE ALSO

`mtskip(1)`, `mtrewind(1)`, `mtforeign(1)`

NOTES

The MCT-6020 hardware will not support transfers of odd numbers of bytes.

The current MCT-6020 firmware (version 1.8) will ignore EOT when executing the `HMT_SKIP` `ioctl`. The tape may be wound off its reel if a filemark is not found.

RTC(7)

UNIX 5.0/5.2

RTC(7)

NAME

rtc - Real Time Clock/Timer driver

DESCRIPTION

Rtc is the Dallas Semiconductor DS1216 "SmartWatch" real time clock device.

/Dev/rtc is used to read or set the RTC. Only one process may have this device open at a time. While the device is open, the process may access the RTC module. There is one ioctl command, RTC_ADRS, which will return the base address of the RTC module.

/Dev/rtcioc1 is used to control the speed of the UNIX clock using an ioctl(2) command. The UNIX clock rate may be adjusted by 1.5%. The following commands are defined in sys/rtc.h.

"CLK_UP" speeds the UNIX clock up.

The device must be kept open for this command to remain effective.

"CLK_DOWN" slows the UNIX clock down.

The device must be kept open for this command to remain effective.

"CLK_RESTORE" returns the UNIX clock to its normal rate.

This command is done automatically if /dev/rtcioc1 is closed while the UNIX clock rate is "abnormal".

The major number for this character device is 15 (M10, V10) or 9 (V20).

/Dev/rtc minor number is 0; /dev/rtcioc1 minor number is 1.

FILES

/dev/rtc

/dev/rtcioc1

INSTALLATION

On the HK68/M10 or V10, the RTC module should be installed in the ODD byte ROM socket, M10-U33 or V10-U23. On the HK68/V20 or M220, install it in U52.

SEE ALSO

clock(1), open(2), read(2), write(2), ioctl(2), close(2)

SCSI(7)

UNIX 5.0

SCSI(7)

NAME

/dev/w[01][a-h] - OMTI-5400 SCSI winchester disks (M10, V10)

DESCRIPTION

This is the driver for the OMTI 5400 disk/floppy/streamer controller when used with the NCR 5380 host controller. Details are in section 17.3.3 of the Heurikon UNIX Reference Guide.

ST(7)

UNIX 5.0

ST(7)

NAME

Notes on the Streaming Tape Driver (M10)

DESCRIPTION

The streaming tape driver user interface is accomplished via two special files. /dev/st1 will rewind whenever it is closed. On writes, /dev/st0 will remain at a position just after the newly written end-of-file mark. If closed during a read (and not yet at an end-of-file mark) /dev/st0 will seek to the next end-of-file mark.

CAVEATS AND NOTES

Tape cartridges are extremely temperature sensitive. Their integrity degrades sharply with increases in temperature.

Tape capacity in megabytes can be approximated by dividing the length of the tape in feet by 10. Thus, a 600 foot tape will have a maximum capacity of 60 Mbytes. Actual capacity will be less due to tape overhead.

It is normal for the drive to stream for one and a half to two seconds and then pause. Each stream represents approximately 130 512 byte buffers being read or written to tape. The Unix file system isn't fast enough to keep a 90 inch per second drive constantly streaming.

Erasing a used tape before reuse helps increase integrity.

Retensioning any tape helps increase integrity on both reading and writing. An erase also retensions the tape.

Erase, rewind, and retension each return control to the user immediately. If another of these commands are executed while a previous one is in progress, the second will not return until the first is complete.

Only one process may have the tape drive open at any time.

DIAGNOSTICS

Error messages will be directed to the system console. They are self-explanatory.

SEE ALSO

stape(1)

NAME

/dev/[n][r]stp[01] - streamer tape (V20)

DESCRIPTION

This is the driver for the streamer tape aspect of the OMTI 5400 disk controller when used with a Plessey host adapter.

The minor number is composed of two parts and can be represented like `^000000r0^`, where `^r^` is the rewind-on-close control bit. If the rewind-on-close bit is set, the tape will not rewind on the last close.

The block size on the tape is fixed at 512 bytes. Attempts to read past end of file will first result in an indication of zero bytes read and further reads will return errors.

IOCTLS

The driver understands one ioctl:

UIOCRETEN Retensions the tape.

SEE ALSO

Heurikon UNIX - Reference Guide

WCH(7)

UNIX 5.2

WCH(7)

NAME

/dev/wch[01][a-h] - SCSI winchester disks (V20)

DESCRIPTION

This is the driver for the OMTI 5400 disk controller when used with a Plessey host adapter.

The minor number is composed of two parts and can be represented like `^0000uppp^`, where `^u^` is the unit designator (drive select) and `^ppp^` is the partition designator. Each drive has eight partitions which are layed out as follows (units in blocks):

<u>Partition</u> (<u>ppp</u>)	<u>Start</u>	<u>Length</u>	
a (000)	34	7966	(swap)
b (001)	8000	b_size	(root)
c (010)	8000	b_size/2	(alt root)
d (011)	8000+b_size/2	b_size/2	
e (100)	34	b_size/2+7966	
f (101)	8000+b_size/2	b_size/4	
g (110)	8000+b_size*3/4	b_size/4	
h (111)	0	maxbn	

Where `maxbn` is the total number of blocks on the disk (i.e. `nsects*nheads*ncyls`) and `b_size` is `maxbn` minus the swap partition size (i.e. 7966) and the number of blocks in the alternate tracks. Approximately 1% of the disk is allocated as alternate track space. The driver uses the controllers bad track mapping facility.

IOCTLS

The driver understands six ioctls:

- UIOCFORMAT** takes a `diskformat` structure as argument, formats the disk, and writes out a disk parameter block (first block on the disk). It currently formats the whole drive.
- UIOCSTEP** takes an array of two chars as argument and changes the controller step rate and step width to the given values (resp).
- UIOCBDBK** forces the driver to reread the disk parameter block (first block on the disk).
- UIOCMAPTRK** takes the address of a block on a bad track and maps that track to the next available alternate.
- UIOCPARTSIZE** returns the size in blocks of the partition.
- UIOCHKFMT** takes the address of a block on a track and returns an error if that track has a defect in it.

SEE ALSO

Heurikon UNIX - Reference Guide

A		C	
accounting system	13,26,80	C compiler	60
adb command	67,76,101	C programming	25
adb command (V.2)	142	C Shell	35,42
adding new users	69	cable connections	153
addresses, memory map	131	calendar program	39
adduser script	72	capacity, floppy	84
administrative functions	1,26,69	cat command	52
alias command	54	CAUTION ITEMS, media	81,85,92
alias examples	35,42,161	CAUTION ITEMS, system	2
alias feature, csh	16,42	cc command	60
aliases, mail	78	cc5.0 command	136
alternate blocks, assignment	100	cd command	28,52
alternate blocks, location	104,105	cdpath variable	50
ampersand character	49	Centronics interface	165
a.out	24,60	character devices	172
as command	61	chgnod	76
at command	38	child process	72
atrun program	38	chstep command	100
awk command	67,185	clicks, memory	122,131
B		clock, baud rate	181
B (Bus) LED	128	clock command	195
background daemons	9,34	clock, cron daemon	38
background tasks (&)	49	coffbin command	67,137
backspace key	170	col command	65
backup, methods	80	command descriptions	23,25
backup script	87	command, examples	52
bad blocks	100	configuration, initial	13
badblk command	95,100	configuration word, Hbug	181
badblk command, V.2	193	console device	32,77,155
bang (!)	42	control characters	46
base addresses	131	core dumps	16,24,73
baud rate clock	181	cp command	53,56
baud rates	4,181	cpio command	82,83,86,88,91,99
baud rates, changing	158	crash, editor recovery	17,58
bcheckrc	34	crash, system	18
big files	74	cron	38,74
block size	84,104	cronlog	39
blocked devices	172	crontab	38
boot, disk location	105	cross-reference, ctags	58
boot diskette	94,98	csh	35
boot procedure, alternate root	6	.cshrc file	36
boot procedure, floppy	12	ctags command	58,63
boot procedure, Winchester	2	cu program	164
boot, writing to disk	100	curly braces {}	43
bootname command	194	curses library	60
Bourne shell, sh	37,42	cut command	87,169
bug reports	20	D	
bus error	6,18	D (DMAC) LED	127
byte swapping, tape	90	daemons	32,34

INDEX

data set, connections	150	error messages	25
data terminal, connections	150	escape character	16,46
Datasud serial card	145	/etc/bcheckrc	34
date command	9,53,74	/etc/brc	34
dd command	83,91,100	/etc/chgnod	76
debuggers	67	/etc/cron	74
default configuration	13	/etc/cshrc	35
DEL key	2,17,170	/etc/gettydefs	33,158,170
density, diskette	84	/etc/init	8,78
/dev directory	41	/etc/inittab	13,32,78,157
/dev/cent	165	/etc/make.smaller	73
/dev/diplied	128	/etc/motd	10,41,72
device drivers	25,41,107,110	/etc/passwd	31
device names/numbers	172	/etc/profile	37
/dev/kmem	109	/etc/rc	13,34,78
/dev/lp	166	/etc/termcap	11,24,36,38
/dev/mem	15,107	/etc/update	129
/dev/st0, /dev/st1	85	/etc/vchk_tree	16
/dev/swap	15,104	Ethernet, file backups	91
/dev/syscon	32,77,155	ex editor	58,59
/dev/systty	32,77,155	EXINIT variable	36,58
/dev/tty	154	exit command	12
df command	72,106	export command	19
dial(3) subroutine	164	exterr command	128
directory structure	27		
disk usage	72	F	
diskconf command	104,142	fdref command	81,196
diskette capacity	84	FIFO (pipes)	48
diskformat command	82,100	file command	53,129
dot (.), dot dot (..)	28	file formats	24
dot files	15	file permissions	15,30
du command	72	file recovery, disk	18,92
dumps, core image	24	file recovery, editor	58
		file system check, fsck	9,102
E		file system check, vchk	76
echo command	54	file system, make	83
ed editor	59	file system organization	29
editor features, ex, vi	57,58	file system repair	30
editor, file recovery	17,58	file system structure	27
editor initialization	58	file types (suffix)	28
editors	59	filename specification	27
egrep command	87	files, hidden	15
end-of-file, EOF	46,170	filters	48
enlarging swap space	101	find command	73,74,86,99
enlarging system parameters	110	fix diskette	103
env command	53	fix diskettes	98
environ variables	19,38,53,75,129	floating point support	127
environmental stabilization	2,3	floppy, booting	12
eqn command	64	floppy capacity	84
error codes	23,124	/floppy directory	12,102
error correction, cmd entry	42,54	floppy diskette operations	81

floppy, error codes	126	I/O errors	124
floppy media	84	I/O system	25
fork(2) system call	50,139	iocheck(), kernel routine	118
formatting, floppies	81	ioctl(2) system call	112,171
formatting, Winchester	92,100	ipcrm and ipcs commands	80,137
FORTTRAN programs, linking	60	ivec.s	114
forwarding mail	78		
fsck command	9,102	J	
fsdb command	30	jumper.c program	108
full file system	72	jumpers, user	128
G		K	
games	24	kernel error messages	25
garbage collection	73	kernel routines	117
gcc command	61	kernel tables	123
getty	33,145	kill command	17,49,72
gettydefs	33,158,170		
gf77 command	61	L	
gobbledygook	31	last command	53
grep command	52	lbolt kernel variable	140
group ownership	30,70	ld command, ROM	135
		ld5.0 command	135,136
H		L-devices file	157,163,164
H (Halt) LED	4,18,128	LEDs, status and user	127,128
Hbug commands	3	libraries, compiler	60
Hbug configuration word	181	line printer scheduler	34,167
head command	54	line printers, hints	165
hex command	135	linked files	56,184
hidden files	15	lint command	63
high density, floppy	84	ln command	56
history command	54	loadsys command	96
history feature	42	loadtar command	96
history variable	50	lockup, terminal	17,18,151
HK68 baud rate, changing	181	log files	74
home directory	27,69	login banner	33
hung program	18	.login file	16,36
		login procedure	10
I		logout command	12
incremental backup	86	.logout file	36
index, permuted	21	lost files, recovery	18,92
init command	77	/lost+found directory	29
init level	32,77,78	lp command	166,169,186
init program	8	LP spooler logic	166
initial configuration	13	lpadmin command	167
inittab	32	lpr program	166
inodes	24	lpsched command	34,167,168
interchange, media	90	ls command	15,52
interrupt, DEL key	2,17,170	lseek(2) system call	141
interrupt service routine	114	L.sys file	157,163
interrupt usage	134		
I/O devices	107		

M		O	
macros, nroff/troff	64	option jumpers	18,128
mail aliases	25,78	output post process	166
mail command	54	owner, file	30
mail command, trouble	16	P	
mail security	79	panic errors	18,130,131
mail variable	50	parent directory	27
major device numbers	172	parent process	72
make command	62	parity, RAM	131
makefile	63	parity, stty command	170
make.smaller script	73	partitions, disk	104
manuals, UNIX	21	Pascal programs, linking	60
map command, editor	58	password, changing/removing	31
maxmeml kernel variable	131	password entry, login	10
mbi7 message	134	password file	31
media, floppy	84	path variable	16,35,50
media interchange	90	pathname	27
memory limitations	131	pattern search, grep	52
memory mapping	108,131	permissions	15,30,76
memory, shared	137	permuted index	21
memory test command, Hbug	4	perror subroutine	139,141,171
message of the day	10,41,72	phys(2) system call	107
messages	137	physical memory map	131
minor device numbers	174	physical shocks	2
mkdir command	28	PID	49,72
mkfs command	83,100	pipes	48
mkfs1b command	83	post process output	166
mknod command	154,176	Power Down procedure	14
mm nroff/troff macros	64	PPID	72
MMU fault	130	pr command	166,169
modem, connections	150	printers, hints	165
modem ports	144	printf subroutine	24,108
modems	162	proc structure	123
more command	52	process number	49
motd file	41,72	profile file	37
mount command	12,34,99,106	prompt, setting	43,50
mounting file systems	34	prompt, user/super-user	11
mtforeign command	90	prompt variable	50
multiple drives	106	ps command	15,52,72
multi-user mode	8,32,74,77	pseudo-users	31
mv command	53,56	psize command	142
N		psize command, V.2	104
namelist, ps command	15	pstat command	80,110
network ports, serial	149	put command, cu	164
new users, adding	69	put command, UniSoft	188
news command	40	pwd command	52
nice command	34,45,49	Q	
nlist() subroutine	140	QUIT interrupt	17
nodename	10,76,162		
nroff	64,183		

INDEX

R		serial port connections	153
RAM size	130	serial ports	144
raw devices	84,172	serial throughput	158
read(2) system call	141	set command	50,53
rebuild diskette	94,98,103	set user id	30
rebuild procedure	92	setenv command	19
Reconfiguration Rights	110	sh	37
redirection of I/O	47	shared memory	80,137
reel-to-reel tape	89	shared text programs	129,136
reference materials	180	shell	42
regular expression	55	shell scripts	16,43
rehash command	16,60	shell variables	50
repair, file system	30	showmatch, editor function	58
repeat command	54	SIGBUS signal	130,131
repeat earlier command	42	signal(2) system call	130,138
reporting bugs	20	SIGSEGV signal	130
reset button	4	single-user mode	8,77,79
reset command	17	sleep command	53,159,164
ring indicator, modems	145	slow boot trouble	8
rm command	42,53	sort command	186
rm, file recovery	18	source code control system	67
rmdir command	28	source code, device drivers	110
ROM programs	135	special characters	46
root & rootcsh logins	10	special files, devices	25
root file system	27,104	spell command	53,58
root user id	31	stack probes	135
RS-232-C	4,145,150	standalone boot prompt	5
rtake command, UniSoft	188	standalone boot, writing	100
run level	77,78	standard input/output/error	47
		static discharges	2
S		status LEDs	127
S (Supervisor) LED	127	sticky bits	129
SANE, stty option	170	stream editor, sed	45,59
sar, system activity package	188	streamer errors	126
SBX-SCC jumpers	182	streamer tape operations	85
SCC clock	181	strings command	16
SCCS	67	stty command	18,35,36,54,158,170
script, big example	185	su command	11,34
script, examples	44,45,72,87,169	subroutines	24
scripts	42,43	SUDH LEDs	127
scroll stop	170	suffix number	23
scrolling display	18	suid	30
sdb command	68	sumdir command	16
search path	16	super-block	14
security	79,146	super-user prompt	8,11
sed command	45,186	swap space	104,129
sed editor	59	swap space, enlarging	101
see command	52	swapping, memory	131
segmentation violation	130	swapping, tape byte	90
semaphores	137	sync command	14
serial expansion	144,152	system administration	1,25,26,69

system calls	23	unalias command	16
system crashes	18	under program	66
system file dumps	99	/unix kernel file	6,76
System III, pgms	90	UNIX manuals	21
system parameters	110	unsupported commands	188
system V.2 notes	142	update program	129
System V.2, porting	136	upper case	10
		Usenet, Users network	40
T		user, adding/removing	69
tabxxx files	66	user id numbers	31
tags feature, editor	58	user jumpers	18,128
tail command	54	user LEDs	128
take command, cu	164	user structure	123
take command, UniSoft	76,188	USERFILE file	163
tape, reel-to-reel	89	/usr/tmp directory	73
tape, streamer operations	85	uucico program	161
tar command	82,83,89,91	uucp modem support	162
tbl command	64	Uucp system	160
telinit command	77		
temporary files	73	V	
TERM (925)	11	variables, shell	50
TERM variable	19	vchk command	76
termcap, entry selection	36	vchk_tree	76
termcap library	24,38,60	vi editor	19,57,59
TERMCAP variable	36	view editor	59
terminal capabilities	24	virtual address, phys call	107
terminal description files	66	virtual memory, /dev/kmem	109
terminal initialization	11	vmail script	45
terminal lockup	17,18,151		
terminal options, stty	18	W	
termination signal	49	w0b	104
terminfo database	38	wait command	49
tilde (~) character	27	wall command	14
time command	54	watchdog timeout	130
time zone, (TZ)	74	wd(0,16000)/unix	102
time/date, setting	9	whereis script	44
/tmp directory	73	who command	53
touch command	87	wild cards	16,43
TPI, floppy tpi	174	Winchester booting	3
tr command	53	Winchester error codes	124
troff	64	Winchester formatting	92
trouble shooting	15	Winchester partitions	104
tset command	36,38,159	Winchester stabilization	5
tty structure	123	working directory	27
tutorials, UNIX manuals	25	write protection, floppy	81
TZ, time zone	74	write protection, reel tape	89
		write protection, streamer	86
U			
U (User) LED	127		
uc command, Hbug	5,182		
umount command	99		

Heurikon Corporation
3201 Latham Drive
Madison, WI 53713
(608) 271-8700