ULTRIX-32 Version 1.2 Release Notes

Order No. AA-BG57C-TE

ULTRIX-32 Version 1.2

Digital Equipment Corporation

DEC	ULTRIX	VAXstation II
DECnet	ULTRIX-11	VAXstation II/GPX
DECUS	ULTRIX-32	VMS
MASSBUS	ULTRIX-32m	VT
PDP	UNIBUS	digital™
Q-Bus	VAX	
RT		

Contents

1 Summary of Features

1.1 New Devices Supported and Devices No Longer Supported 1-1

1.2 Documentation Improvements 1-1

1.2.1 Installation Guide Organization Improved 1-1
1.2.2 Programmer's Manual Pages Added and Modified 1-2
1.2.3 System Manager's Guide Expanded 1-2

1.3 New Commands, Utilities, and Features 1-2

1.3.1 rabads Stand-Alone Bad Block Replacement Program 1-2
1.3.2 Internet Subnet Routing Supported 1-2
1.3.3 Internet Daemons Combined (inetd) 1-2
1.3.4 New arff Options and Modifiers 1-3
1.3.5 New tar and mdtar Options 1-3
1.3.6 Improved Format for /etc/ttys File 1-3
1.3.7 Terminal Lines Shared for tip and uucp 1-3
1.3.8 New Configuration File Option (EMUFLT) 1-3
1.3.9 Accounting Accuracy Increased (acct) 1-4
1.3.10 Improvements to Standard I/O System (stdio) 1-4
1.3.11 Symbolic Debugger (dbx) Supported 1-5

1.3.11.1 New Features 1-5
1.3.11.2 Problems Fixed 1-6

1.4 System V Feature Compatibility 1-7

1.4.1 Using the System V Features 1-7
1.4.2 System V System Services and Library Routines 1-8

1.4.2.1 System V Base System Services 1-8
1.4.2.2 System V Base Library Routines 1-13

1.4.3 System V Kernel Extensions 1-16

2 Notes on Software

2.1 Corrections to ULTRIX-32 Installation Guide 2-1

2.2 Corrections to the ULTRIX-32 System Manager's Guide 2-2

2.3 Corrections to the ULTRIX-32 Programmer's Manual 2-2

2.4 Important Notices Pertaining to the Installation 2-4

2.4.1 Back Up User-Altered Files 2-4
2.4.2 Installation Script Fails to Recover 2-5
2.4.3 Correcting Errors That Occur When Checking the File System 2-5
2.4.4 /usr File System Full If All Subsets Loaded 2-5
2.4.5 Time Zones East of GMT Mishandled During Installation 2-6
2.4.6 Setting Up Local Area Transport (LAT) Facilities 2-7
2.4.7 Pseudoterminals May Become Unusable After System Crash 2-7
2.4.8 uucpsetup Does Not Set Up /etc/remote 2-8
2.4.9 uucpsetup Does Not Set Up tty Shared Mode Lines 2-8

2.5 Known Problems and Corrections 2-8

2.5.1 Halting a Running VAX 8600 or VAX 8650 Processor Can Cause Failures 2-8
2.5.2 Hardwired Terminals May Not Respond After Being Turned Off 2-9
2.5.3 Tape Drive May Hang on Remote System If Local System Crashes 2-10
2.5.4 instmedia Causes Error After Use 2-11
2.5.5 netsetup Command May Ask for System Name 2-11
2.5.6 uucpsetup Installs Modem Line on ttyh2 2-11
2.5.7 uucp and tip May Not Work with DMF32 2-12
2.5.8 uucp May Not Work After Update From Version 1.1 2-12
2.5.9 Do Not Make Links to /dev/tty Devices 2-13
2.5.10 /usr/adm/aculog Mode May Cause Security Problem 2-14
2.5.11 C Optimizer Fails 2-14

2.5.12 f77 Fails To Resolve All External Symbols 2-15
2.5.13 Entry in /etc/acucap Does Not Work for Ventel Modems 2-15
2.5.14 Interaction Between DECnet and arp Causes Network Problem 2-16
2.5.15 script Command Interprets CTRL/Z and CTRL/C Signals 2-16
2.5.16 Improper Use of sccs fix Command Leaves File Unusable 2-17
2.5.17 beautify Variable in tip Can Not Be Unset 2-17
2.5.18 fortunes.dat File Has Wrong Permission Modes Set 2-17
2.5.19 On-Line Documentation for mtio is Incomplete 2-17

2.6 General Notes 2-18

2.6.1 Capacity Upgrade Kits from Version 1.1 Do Not Work 2-18
2.6.2 New Format for /etc/ttys File 2-18
2.6.3 Some User Programs Must Be Recompiled 2-18
2.6.4 Default Filter Needed in Printcap File for LP11 Printers ... 2-18
2.6.5 Spool Directories for Remote Printers 2-19
2.6.6 SO_DONTLINGER Option No Longer Exists. 2-19
2.6.7 SO_BROADCAST Option Added 2-19
2.6.8 dskx May Corrupt Disk or Destroy Superblock 2-19

A Configuring the MICOM PAD for tip and uucp

A.1 Description of the MICOM PAD A-1

A.2 Connecting the PAD A-1

A.3 Setting Up uucp for the PAD A-2

A.3.1 Setting Up Incoming uucp Lines A-3
A.3.2 Setting Up uucp to Dial Out Over the PAD A-4

A.4 Setting Up tip for Use with the PAD A-6

A.4.1 Setting Up Outgoing tip Connections A-6
A.4.2 Setting Up Incoming login Service A-8

B New and Replacement Programmer's Manual Pages

Organization of These Notes

These Release Notes apply to the ULTRIX-32 Version 1.2 software kit and documentation set.

Read these Notes **before** you install your distribution kit software. If you discover errors, omissions, or inaccuracies as you use the software or this documentation, please submit a Software Performance Report (SPR) or Reader's Comment form. Your comments and criticisms help us improve the product.

These Release Notes have two chapters and two appendixes:

Summary of Features

Describes the changes and additions included in this release.

Notes on Software

Contains descriptions of changes made too late to be included in the documentation set.

Also includes important notices that you must read before installing or using your system. Certain system features or the system itself may not work unless you make the changes indicated in this chapter.

Configuring the MICOM PAD for tip and uucp

Describes how to configure a MICOM[1] Micro800/X.25 PAD for use with the tip(1) and uucp(1) utilities.

New and Replacement Programmer's Manual Pages

Contains new and replacement pages for the *ULTRIX-32 Programmer's Manual.*

[1] MICOM is a trademark of MICOM Systems, Inc.

Summary of Features 1

ULTRIX-32 Version 1.2 is an update release that includes the following changes and new functionality:

- New device support
- Documentation improvements
- New commands, utilities, and features
- System V feature compatibility

This release supersedes ULTRIX-32 Version 1.1.

1.1 New Devices Supported and Devices No Longer Supported

This release supports the following processors and new devices:

- VAX 8200, VAX 8600, and VAX 8650 processors
- TSU05 magnetic tape drive
- DF224 2400-baud modem

Distribution kits for full installations of the ULTRIX-32 operating system are no longer available on RL02 disks. This release is the last for which RL02 update installation distribution kits will be available.

The VAX-11/725 processor is no longer supported.

1.2 Documentation Improvements

All of the documents in the set are revised and expanded to include new and changed material.

1.2.1 Installation Guide Organization Improved

The material in Chapter 3 of the *ULTRIX-32 Installation Guide* is streamlined to help the installer quickly find the information needed to install the software on a particular hardware configuration.

1.2.2 Programmer's Manual Pages Added and Modified

The *ULTRIX-32 Programmer's Manual* includes additional pages to describe subroutines and system calls that have been added or modified to add System V feature compatibility. Other pages also have been added or modified to reflect new or changed software features.

See Appendix B for new Programmer's Manual pages that were added too late for inclusion in the Programmer's Manual.

1.2.3 System Manager's Guide Expanded

The *ULTRIX-32 System Manager's Guide* includes new chapters and appendixes, including information on using local area transport (LAT) facilities with the DECserver 100 terminal server.

1.3 New Commands, Utilities, and Features

The following sections describe commands, utilities, and features added in this release.

1.3.1 rabads Stand-Alone Bad Block Replacement Program

This release contains the latest version of rabads, a stand-alone program that finds and replaces bad blocks on DIGITAL Storage Architecture (DSA) disks that use UDA-type and RQDX-type controllers. See the *ULTRIX-32 System Manager's Guide* for a complete description of bad block replacement strategies and how to use rabads.

1.3.2 Internet Subnet Routing Supported

This release supports Internet subnet routing. The *ULTRIX-32 System Manager's Guide* describes this functionality in the context of local area networks.

1.3.3 Internet Daemons Combined (inetd)

In previous releases, all Internet daemons were separate; their heavy demands on system resources resulted in overloading small or heavily loaded systems.

The inetd daemon now acts as a "listening agent" for all Internet daemons. The configuration file /etc/inetd.conf lists the standard services provided by inetd. The system manager can alter this file to add nonstandard services.

See inetd(8c) and inetdconf(5) in the Programmer's Manual for more information.

1.3.4 New arff Options and Modifiers

The arff(8v) command includes several new options and modifiers and a change in default behavior.

The following are the new options and modifiers:

New Options		New Modifiers	
–i	initialize	–h	ignore home block corruption
–p	protect	–p	file is printable
–u	unprotect	–n	noextract

In addition, the –b option (write boot block) now supports VAX 8600 and VAX 8650 console media.

The default file name of the file containing the RT-11 device image is now /dev/bootdev (formerly /dev/floppy).

For more information, see arff(8v) in the Programmer's Manual.

1.3.5 New tar and mdtar Options

The tar(1) and mdtar(1) commands have new functions and options. See the mdtar(1) and tar(1) replacement pages in Appendix B for further information.

1.3.6 Improved Format for /etc/ttys File

The format of the /etc/ttys file has improved. It has been merged with the /etc/ttytype file and is now easier to understand and modify. See tty(5) for further information.

1.3.7 Terminal Lines Shared for tip and uucp

This release supports shared terminal lines. This feature allows you to use the same lines for incoming (login) and outgoing (tip, uucp) connections without reassigning terminal lines.

1.3.8 New Configuration File Option (EMUFLT)

Entering the EMUFLT option in the system configuration file creates floating point emulation in software for processors that do not include hardware floating point instructions.

1.3.9 Accounting Accuracy Increased (acct)

The accounting facility records more reliable accounting data. See acct(5) in the Programmer's Manual for more information.

1.3.10 Improvements to Standard I/O System (stdio)

The following lists the improvements to the standard I/O system (stdio) for this release:

- FILE structures for files other than stdin, stdout, and stderr are allocated dynamically using malloc(3c) when fopen(3s) or fdopen(3s) are called. This change may affect programs that assume FILE structures for stdio are allocated as a static array at compile time in one or both of the following ways:
  - Previously, some programs failed to explicitly keep track of open files. To close open files, they used information about the internal details of the stdio system (specifically that FILE structures were stored as an array). To be sure all files were closed, they stepped through the array checking for open files, and then closed those files that indicated they were open.

    This and similar techniques were highly dependent on the internal details of the stdio system. Because the internal details of stdio have changed in this respect, these techniques no longer work. The correct method to handle this problem is for programs to explicitly keep track of all open files.
  - File structures being allocated dynamically may interfere with programs that have their own memory allocation scheme that can not coexist with malloc(3). The principle characteristic of such a memory allocation scheme is that it assumes all calls to sbrk(2) result in contiguous sections of memory being allocated. There are two work-arounds to this problem:
    1. Redesign the program's private memory allocation scheme so that it does not depend on all calls to sbrk(2) returning memory contiguous with memory allocated by previous calls to sbrk. Using malloc(3) is one solution.
    2. Make sure that all files have been opened and any associated buffers (see below) allocated before the private memory allocation system is used.

- The buffers for any buffered file, including stdin and stdout, are allocated dynamically using malloc(3) unless buffer allocation is explicitly done by setbuf(3s) or setubuf(3s).

- Using %X in a format control string for printf(3s), fprint(3s), or sprintf(3s) causes the corresponding argument to be printed as a hexadecimal number using uppercase letters for the alphabetic hexadecimal digits (A-F). Using %x causes these letters to be printed in lowercase.
- scanf(3s), fscanf(3s), and sscanf(3s) now behave as follows: if the scan fails to find a field using the %[conversion in the control string, the scan returns immediately with the number of items found up to that point.

1.3.11 Symbolic Debugger (dbx) Supported

The dbx(1) command, a source-level symbolic debugger supporting multiple languages, now contains new features and problem fixes. The following sections briefly describe the changes to dbx.

1.3.11.1 New Features – Listed here are either new features or those not described in the dbx(1) Programmer's Manual:

- The rerun command reruns a program using the arguments specified with the previous run command.
- The return [*procedure*] command continues until a return to the argument *procedure* is executed, or until the current procedure returns if no argument is specified.
- The up [*count*] command moves the current scope, which is used for resolving names, up the stack the number of levels specified in the argument *count*. If no argument is specified, the default number of levels is one.
- The down [*count*] command is similar to the up command except that the down command moves the specified number of levels down the stack.
- The cont [*n*] command accepts a signal on the command line; *n* is an integer representing a signal, and the allowable range is 1 to 32 (see the signal.h header file).
- Under the header Machine Level Commands in the dbx(1) Programmer's Manual page is a description of a method of displaying the contents of memory. The method described requires a numerical address in hexadecimal, octal, or decimal or a symbolic address in the form &symbol.

  You also can display the contents of a symbolic address by enclosing the symbol in parentheses. For example, if the symbol ptr contains the address 0x100, then either of the following two commands print the 10 instructions starting at 0x100:

```
0x100/10 i

(ptr)/10 i
```

- The delete command permits commas on the line; for example:

  ```
  delete 2,3,5
  ```

- The delete * command removes all existing breakpoints and tracepoints at once.
- dbx saves the most recent command line. Pressing the RETURN key by itself on a command line causes the previously executed command to be executed again. You can use this feature for single-stepping through a program.

  This feature may not behave as expected when examining memory, however; because the current location pointer is not updated, the same location will be displayed again. Instead, to examine consecutive memory locations, use the following sequence:

  ```
  0x100/i
  /i                      (current location pointer)
  <RETURN>
  <RETURN>
       .
       .
       .
  ```

- The init file (normally .dbxinit in your home directory) is built by appending the characters init to the first eight characters of the debugger's name. For example, if you renamed dbx to abc, the debugger would look for a file named .abcinit in your home directory. If you renamed the debugger to This_is_dbx, the debugger would look for an initialization file named .This_is_init in your home directory.

1.3.11.2 Problems Fixed – The following problems that existed in previous versions of dbx are fixed in this release:

- The previous release included two versions of dbx: dbx and mdbx. mdbx was the updated version of dbx and included new features. But mdbx did not work with Fortran, so the unaltered version of dbx was included for Fortran debugging. The version of dbx included in this release is an improved version of mdbx; you no longer need (and should delete remaining copies of) mdbx.

- There is no longer a limit on the maximum number of functions allowed in the program being debugged.
- The trace command now works with recursive routines.

1.4 System V Feature Compatibility

This release includes features that are compatible with those contained in UNIX System V software.

The features include System V Interprocess Communication (IPC) and system call interface, and library modifications. The IPC facilities include:

- Shared memory — allows variable-sized segments of data memory to be shared between unrelated processes
- Semaphores — permits synchronization between unrelated processes
- Message queues — permits messages to be sent between unrelated processes
- Named pipes — permits pipelining between unrelated processes

The system call interface and library modifications include the following two new libraries:

- libcV.a — C library
- libmV.a — math library

1.4.1 Using the System V Features

Most of the System V features are library calls available for use in any ULTRIX-32 program.

Some of the System V system calls and library routines added in this release, however, conflict with those already present in the ULTRIX-32 system in that they are called or behave differently. These features are not contained in the standard C runtime library. Instead, you can choose either the regular ULTRIX-32 behavior of these features or System V-compatible behavior. Choosing the System V behavior allows you to port applications developed on UNIX System V to the ULTRIX-32 system.

When you specify the System V environment when you compile and link programs, the conflicting calls and routines are compatible with System V; otherwise, they behave in the normal ULTRIX-32 (and Berkeley 4BSD) manner. You can specify the System V environment in either of the following ways:

- Include the -Y option when you use the cc(1) command.

- Set the environment variable PROG_ENV to SYSTEM_FIVE (see sh(1) or csh(1) for information on setting environment variables).

Using either of these methods causes cc(1) to do the following:

- Define the preprocessor symbol SYSTEM_FIVE, causing cpp(1) to select the System V versions of various data structures and symbol definitions.
- If cc(1) invokes ld(1), cause libcV.a (the System V version of the standard C library) to be searched to resolve references to the System V-specific routines. If you include the -lm options on the command line, the System V version of the math library (libmV.a) is used instead of the regular ULTRIX-32 math library.

1.4.2 System V System Services and Library Routines

The following sections describe the system services and library routines added or modified in this release to provide System V feature compatibility. The entries in each section describe the degree to which the service or routine conforms to the *System V Interface Definition.*[1] Unless otherwise specified, these services and routines are also available with the same behavior in the ULTRIX-32 environment.

1.4.2.1 System V Base System Services – This section lists the System V base system services provided in this release. Each service listed conforms with the System V definition unless otherwise specified.

abort	generate an abnormal process abort Conforms to the System V definition, but differs from the ULTRIX-32 version in that it attempts to close any open files before aborting.
access	determine accessibility of a file
alarm	set a process alarm clock
chdir	change working directory Differs from the System V definition in that ELOOP is a possible error condition.
chmod	change mode of file

[1] The *System V Interface Definition* is published by and available from AT&T. It is not included in the ULTRIX-32 documentation set.

	Differs from the System V definition in that the mode bit 02000 (set group identification on execution) is not cleared if the effective user identification of the process is not superuser or the effective group ID of the process does not match the group identification of the file. Also, ELOOP is a possible error condition.
chown	change owner and group of a file Differs from the System V definition in that only the superuser may change the ownership. Also, ELOOP is a possible error condition.
close	close a file descriptor
creat	create a new file or rewrite an existing one Differs from the System V definition in that ELOOP and ENXIO are possible error conditions.
dup	duplicate an open file descriptor
exec	execute a file (execl, execv, execle, execve, execlp, execvp)
exit	terminate a process (exit, _exit) Differs from the System V definition in that even if the calling process is a process group leader, the SIGHUP signal is not sent to each process that has a process group identification equal to that of the calling process.
fclose	close or flush a stream (fclose, fflush)
fcntl	file control
ferror	stream status inquiries (ferror, feof, clearerr, fileno)
fopen	open a stream (fopen, freopen, fdopen)
fork	create a new process
fread	binary input/output (fread, fwrite)
fseek	reposition a file pointer in a stream (fseek, rewind, ftell)
getcwd	get pathname of current working directory This has also been added to the ULTRIX-32 environment.

getpid

get process, process group, and parent process identifications (getpid, getpgrp, getppid)

Differs from the System V definition in that the return values for getpid and getppid are longwords instead of integers. In the System V environment, getpgrp is called with no arguments, which is the same as calling getpgrp with 0 as the argument in the ULTRIX-32 environment.

getuid

get real user, effective user, real group, and effective group identifications (getuid, geteuid, getgid, getegid)

Differs from the System V definition, in that the return values are integers instead of unsigned shorts.

ioctl

control device

The System V ioctl(2) emulation maps TERMIO ioctl calls onto ULTRIX-32 TTY ioctl calls. It handles the differences in the System V and ULTRIX-32 sgttyb structure. Note that this is not the full implementation of TERMIO. It maps those functions that are common to ULTRIX-32 and System V software.

kill

send a signal to a process or a group of processes

Differs from the System V version when PID equals −1 and the effective user identification of the sender is not the superuser. In this case a signal should be, but is not, sent to all processes, excluding special system processes, whose real user identification is equal to the effective user identification of the sender.

The System V version provides the same functionality as the ULTRIX-32 kill(2) and killpg(2) system calls combined.

link

link to a file

lockf

record locking on files

Differs from the System V definition in that EOPNOTSUPP is a possible error condition.

lseek

move read/write file pointer

Conforms to the System V definition, but differs from the ULTRIX-32 version in that it sends a SIGSYS signal in addition to setting *errno*, if the "whence" argument is invalid.

malloc	fast main memory allocator (malloc, free, realloc, calloc, mallopt, mallinfo) The new functions mallopt and mallinfo are available to regular ULTRIX-32 programs when linked with the malloc library. (See the malloc.h header file and use –lmalloc when linking.)
mknod	make a directory, or a special or ordinary file Named pipes (FIFOs) have been added to the ULTRIX-32 software.
mount	mount a file system Conforms to the System V definition, but differs from the ULTRIX-32 version in the value of *errno* on errors as follows: • ENOTDIR instead of EPERM (illegal characters in directory name) or EROFS (directory on read-only file system) • EPERM instead of ENODEV if not superuser; ENOENT instead of ENODEV if the file specification is nonexistent
open	open for reading or writing Conforms to the System V definition, but differs from the ULTRIX-32 version in that O_NDELAY may affect subsequent reads and writes.
pipe	create an interprocess channel
popen	initiate pipe to/from a process
read	read from file Conforms to the System V definition, but differs from the ULTRIX-32 version in that it does not return an error if O_NDELAY is set and no data is ready.
setpgrp	set process group identification In the System V environment, setpgrp is called with no arguments, which is the same as calling setpgrp with 0 as the argument in the ULTRIX-32 environment.
setuid	set user and group identifications (setuid, setgid)

	Differs from the System V definition in that it returns EPERM instead of EINVAL if the argument is out of range.
signal	specify what to do on receipt of a signal Conforms to the System V definition, but differs from the ULTRIX-32 version in the following ways: • The handler function does not remain installed after the signal has been delivered. • When a signal to be caught occurs during a read, write, open, or ioctl to a slow device (like a terminal, but not a file) or during a pause or wait, the signal handler function will be executed, and then the interrupted system call may return a −1 to the calling process with *errno* set to EINTR.
sleep	suspend execution for interval
stat	get file status (stat, fstat) Differs from the System V definition only in that EPERM and ELOOP are also possible error conditions.
stime	set time
sync	update superblock
system	issue a command
time	get time
times	get process and child process times. Conforms to the System V definition, but differs from the ULTRIX-32 version in that it returns the elapsed time in seconds since system startup (instead of zero) on success.
ulimit	get and set user limits
umask	set and get file creation mask
umount	unmount a file system Conforms to the System V definition, but differs from the ULTRIX-32 version in that it returns EPERM instead of ENODEV if not superuser, and ENXIO instead of ENODEV if the special file does not exist.

uname	get name of current UNIX system This is also available in the ULTRIX-32 environment.
unlink	remove directory entry Differs from the System V definition in that ETXTBSY is not a possible error condition, but ELOOP is.
ustat	get file system statistics
utime	set file access and modification times
wait	wait for child process to stop or terminate Conforms to the System V definition, but differs from the ULTRIX-32 version in that wait waits until all children terminate, if the SIGCLD (maps to SIGCHLD on ULTRIX-32 software) signal is being ignored.
write	write on a file See the description of the read routine for discussion of O_NDELAY handling.

1.4.2.2 System V Base Library Routines – This section lists the System V base library routines provided. Each library routine listed conforms with the System V definition unless otherwise specified.

abs	return integer absolute value
assert	verify program assertion
bessel	Bessel functions (j0, j1, jn, y0, y1, yn)
bsearch	binary search a sorted table
clock	report CPU time used
conv	translate characters (toupper, tolower, _toupper, _tolower, toascii)
crypt	generate DES encryption (crypt, setkey, encrypt)
ctermid	generate file name for terminal
ctime	convert date and time to string (ctime, localtime, gmtime, asctime, tzset) Conforms to the System V definition, but differs from ULTRIX-32 environment in terms of time zone handling.

ctype	classify characters (isalpha, isupper, islower, isdigit, isxdigit, isalnum, isspace, ispunct, isprint, isgraph, iscntrl, isascii)
drand48	generate uniformly distributed pseudorandom numbers (drand48, erand48, lrand48, nrand48, mrand48, jrand48, srand48, seed48, lcong48)
erf	error function and complementary error functions (erf, erfc)
exp	exponential, logarithm, power, and square root functions (exp, log, log10, pow, sqrt)
floor	floor, ceiling, remainder, absolute value functions (floor, ceil, fmod, fabs)
frexp	manipulate parts of floating point numbers
ftw	walk a file tree
gamma	log gamma function
getc	get character or word from a stream (getc, getchar, fgetc, getw)
getenv	return value for environment name
getopt	get option letter from argument vector
gets	get a string from a stream (gets, fgets)
hsearch	manage hash search tables
hypot	Euclidean distance function
lsearch	linear search and update (lsearch, lfind)
matherr	error-handling function
memory	memory operations (memccpy, memchr, memcmp, memset)
mktemp	make a unique file name
perror	system error messages (perror, errno, sys_errlist, sys_nerr)
printf	print formatted output (printf, fprintf, sprintf)
putc	put character or word on a stream (putc, putchar, fputc, putw)

putenv	change or add value to environment
puts	put a string on a stream (puts, fputs)
qsort	quicker sort
rand	simple random number generator (rand, srand)
regcmp	compile and execute a regular expression (regcmp, regex)
scanf	convert formatted output (scanf, fscanf, sscanf)
setbuf	assign buffering to a stream
setjmp	nonlocal goto (setjmp, longjmp)
sinh	hyperbolic functions (sinh, cosh, tanh)
ssignal	software signals (ssignal, gsignal)
string	string operations (strcat, strncat, strcmp, strncmp, strcpy, strncpy, strlen, strchr, strrchr, strpbrk, strspn, strcspn, strtok)
strtod	convert string to double-precision number (strtod, atof)
strtol	convert string to integer (strtol, atol, atoi)
swab	swap bytes
tmpfile	create a temporary file
tmpnam	create a name for a temporary file (tmpnam, tempnam)
trig	trigonometric functions (sin, cos, tan, asin, acos, atan, atan2)
tsearch	manage binary tree search (tsearch, tfind, tdelete, twalk)
ttyname	find name of a terminal (ttyname, isatty)
ungetc	push character back into input stream
vprintf	print formatted output of a varargs argument list (vprintf, vfprintf, vsprintf)

1.4.3 System V Kernel Extensions

This section describes the System V kernel extensions provided. Each extension listed conforms with the System V definition unless otherwise specified.

acct	enable or disable process accounting Differs from the System V definition in that ELOOP is a possible error condition. Also, no error is indicated if an attempt is made to enable accounting when it is already enabled.
chroot	change root directory Differs from the System V definition in that EACCES and ELOOP are possible error conditions.
msgctl	message control operations
msgget	get message queue
msgop	message operations
nice	change priority of a process
plock	lock process, text, or data in memory
profil	execution time profile
ptrace	process trace
semctl	semaphore control operations
semget	get set of semaphores
semop	semaphore operations
shmctl	shared memory control operations
shmget	get shared memory segment
shmop	shared memory operations

Notes on Software 2

This chapter discusses known problems with the software documentation and the ULTRIX-32 software. Read this chapter **before** you install or use the ULTRIX-32 software distribution kit.

Here are the topics discussed in this chapter:

- Corrections to the *ULTRIX-32 Installation Guide*
- Corrections to the *ULTRIX-32 System Manager's Guide*
- Corrections to the *ULTRIX-32 Programmer's Manual*
- Important notices pertaining to the installation
- Known problems and corrections
- General notes

Note

To execute most of the commands in this chapter, you must have superuser privileges. It is best to use the su(1) command to temporarily substitute the superuser (root) identification for your own.

2.1 Corrections to ULTRIX-32 Installation Guide

Make the following changes to the *ULTRIX-32 Installation Guide:*

- Throughout the Guide, all references to the VAX 8600 processor also apply to the VAX 8650 processor.
- Page 7-7: The text refers to the file /usr/lib/lpr as a directory. It is a file.
- Page 3-48, Step 10 (Load the Generic System): Add ra60 to the list of recognized distribution media devices.
- Page 5-2, Step 3 (Select from the Menu): The list of software subsets shown in the example may not match the list the installation procedure prints on your terminal. Use the list shown on your terminal to select the software subsets you want to install.
- Page 4-5: The text states that during Phase 2 of the installation procedure, you are prompted to verify your choice of the accounts

directory. This prompt does not appear if you accept the default.

2.2 Corrections to the ULTRIX-32 System Manager's Guide

Make the following corrections to the *ULTRIX-32 System Manager's Guide*:

- Throughout the Guide, all references to the VAX 8600 processor also apply to the VAX 8650 processor.
- Page A-6, Appendix A: The third paragraph states that the description of changes you enter must be on one line, and that you can add more lines by ending each line except the last with a backslash. This is incorrect. You can enter one or more lines of text, including RETURN characters, and you do not need to include backslash characters at the end of each line. When you finish entering your comments, exit from the text entry mode by typing CTRL/D at the beginning of a new line.

2.3 Corrections to the ULTRIX-32 Programmer's Manual

Make the following corrections to the *ULTRIX-32 Programmer's Manual*:

recv(2) The line **struct msghdr msg[ ];** in the SYNTAX section for the **recvmsg** call should read:

struct msghdr *msg[ ];

send(2) The line **struct msghdr msg[ ];** in the SYNTAX section for the **sendmsg** call should read:

struct msghdr *msg[ ];

ustat(2) The line **int dev;** in the SYNTAX section of the ustat(2) manual page should read:

dev_t dev;

dhv(4) The first line of the FILES section should read:

```
/dev/tty[h-i][0-7]
```

dz(4) The dz(4) page should list the DZ32 communications multiplexer in the SYNTAX line, in addition to the DZ11 communications multiplexer.

tmscp(4) The tmscp(4) manual page should list the TU81 tape drive in addition to the TK50 magnetic tape interface.

ts(4) The ts(4) page should list the TU80 tape drive in addition to the TS11 magnetic tape interface.

tty(4) The TAB1 value for the *sq_flags* field should be 0002000 instead of 0001000.

ttys(5) The ttys(5) manual page incorrectly states that if the flags modem or nomodem are not specified, the default flag is nomodem.

The default is modem.

fortune(6)

- Under the description of the -w flag, the word Waits should read Wait.
- Add the following to the DESCRIPTION section:

  There is only one fortune in the file /usr/games/lib/fortunes.dat.

  To add fortunes, put them in a file with each fortune separated by a line that contains only two percent signs (%). Use that file as an argument to /usr/games/lib/strfile, which produces a file named fortunes.dat. Put the fortunes.dat file in /usr/games/lib with read permissions set.

lcp(8) The lcp(8) command has a -g option that can be used with any of the other lcp options. Change the first line of the SYNTAX section to read:

lcp [**-sr**] [**-g** *grp1, grp2, ...*] [**-n** *node*]

Add the following description of the **-g** option:

-g Set groups to *grp1, grp2, ... , grpn.*

Some Section 3c Pages Are Incorrectly Named

Some pages in Section 3c of the Programmer's Manual should be in Section 3.

For the ULTRIX-32 documentation set, libraries described in Section 3c are routines included for compatibility with other systems. In particular, a number of system call interfaces provided in previous releases of 4BSD have been included for source code compatibility. The Programmer's Manual entry for each compatibility routine indicates the proper interface use.

For the UNIX System V documentation set, libraries described in 3c are for C and assembler library routines.

Move the following pages from Section 3c to Section 3:

a64l	hsearch
alarm	l3tol
bsearch	lsearch
clock	memory
conv	nice
drand48	putenv
ftok	putpwent
ftw	strtod
getcwd	strtol
getopt	time
getut	times
	tsearch

2.4 Important Notices Pertaining to the Installation

The notes in this section pertain to the installation procedure. Read these notes before you begin to install the ULTRIX-32 operating system.

2.4.1 Back Up User-Altered Files

Before you begin the installation, use the tar(1) command to copy any files you have altered or created in the file system, including all the files in user directories.

2.4.2 Installation Script Fails to Recover

The installation procedure fails to detect devices that are inaccessible. For example, if the default tape drive is off line, the installation procedure continues, even though no data is being read from the distribution tape.

To avoid this problem, be sure your default distribution device is on line before you begin the installation.

2.4.3 Correcting Errors That Occur When Checking the File System

The fsck(8) command checks the file system at the following stages of the ULTRIX-32 installation:

- Phase 1, Step 10 for VAX-11/730, VAX-11/750, VAX-11/780, VAX-11/785, VAX 8600, and VAX 8650 processors
- Phase 1, Step 11 for VAX 8200 processors
- Phase 2, Step 4 (for all processors)

In most installations, fsck completes checking the file system without discovering any errors. If fsck displays errors during any of the previously mentioned steps of the installation, then either the system disk or the distribution media is corrupt.

To ascertain why fsck displays errors, do the following:

1. Replace the disk pack that is currently the system disk with another disk pack.
2. Start the installation over again, beginning with Phase 1, Step 1.
3. If fsck does not display any errors, then the original system disk was corrupt. Call your DIGITAL representative to reformat the corrupt disk pack.

   If fsck continues to display errors, then the problem is probably with the distribution media. Contact your DIGITAL representative to replace the corrupt distribution media.

2.4.4 /usr File System Full If All Subsets Loaded

If you load all of the software subsets in Phase 3 of the installation, the /usr file system does not have sufficient space remaining to accommodate user directories and files. You should choose and load only the software subsets that you actually intend to use on your system.

2.4.5 Time Zones East of GMT Mishandled During Installation

The ULTRIX-32 installation procedure does not correctly handle time zones east of Greenwich, England and west of the International Date Line.

To correct this situation, follow these steps:

1. Proceed with the installation and tell the installation program that you are using Greenwich Mean Time (GMT). Enter the GMT date and time when prompted for them.
2. After the installation is complete, but before you install any layered products, read Chapter 8 of the *ULTRIX-32 System Manager's Guide* for information on building the kernel.
3. Edit the configuration file in /usr/sys/conf. The name of the configuration file on your system is the name of the system spelled in capital letters (for example, MYVAX). Change the time zone parameter in the configuration file. The current value is 5, and the line is set up to use daylight savings time (DST). If you are in the GMT zone, but prefer the designation Western European Time (WET) zone or DST, then replace the 5 with 0. If you are in the Middle European Time (MET) zone, use −1, and if you are in the Eastern European Time (EET) zone, use −2. If your location does not use daylight savings time, then remove the dst.

   You can specify time zones other than those supported and mentioned above by using other negative values for the time zone parameter. For values less than −3, though, the system will print a time zone abbreviation relative to GMT, such as GMT+4, instead of the actual abbreviation for the time zone.
4. Rebuild the system as described in the System Manager's Guide.
5. Edit the /etc/rc file. In this file you will find the following lines:

```
tz='cat /etc/timezone'
dt='date -u "+${PERCENT}y${PERCENT}m${PERCENT}d${PERCENT}H${PERCENT}M"'
date -u "$dt-$tz"
```

   Change these lines to:

```
#tz='cat /etc/timezone'
#dt='date -u "+${PERCENT}y${PERCENT}m${PERCENT}d${PERCENT}H${PERCENT}M"'
#date -u "$dt-$tz"
```

6. Reboot the system. One way of rebooting the system is to type:

```
# /etc/shutdown -r now
```

The reboot(8) command shuts down the system, then reboots it with the new copy of the kernel.

2.4.6 Setting Up Local Area Transport (LAT) Facilities

This release supports the local area transport (LAT) remote node maintenance functions that are used with terminal servers such as the DECserver 100.

The terminal server software that is downloaded into the DECserver 100 is supplied with the terminal server hardware, not with the ULTRIX-32 software. The ULTRIX-32 system does include software for the remote node maintenance functions. See Chapter 9 of the *ULTRIX-32 System Manager's Guide* for information on using these functions to load and control terminal servers. See also the descriptions of the individual maintenance commands, control program, and driver in the *ULTRIX-32 Programmer's Manual:*

lta(4)
addnode(8)
ccr(8)
getnode(8)
lcp(8)
load(8)
mop_mom(8)
remnode(8)
trigger(8)

For information on how to set up local area transport (LAT) lines in the /etc/ttys file, see the ttys(5) description in the *ULTRIX-32 Programmer's Manual.*

You should place an entry in the /etc/rc.local file on your system to run /etc/mop_mom as a background task.

2.4.7 Pseudoterminals May Become Unusable After System Crash

If a remote connection to a pseudoterminal is broken unexpectedly (for example, if the system crashes), the ownership and modes of the pseudoterminal special file may be set incorrectly. This can cause some utilities to be unable to access the pseudoterminal and fail.

Adding the following lines to the end of your system's /etc/rc.local file causes the ownership and modes of all pseudoterminal special files to be reset correctly when the system is rebooted:

```
/etc/chown root /dev/*ty[p-u]*
/bin/chmod 666 /dev/*ty[p-u]*
```

2.4.8 uucpsetup Does Not Set Up /etc/remote

The uucpsetup command does not modify the /etc/remote file to contain the type of modem you have connected to the ttyd2 line. Therefore, if you want to use tip(1) and your modem is not a DF03, you must edit the /etc/remote file and change all occurrences of df03 to the appropriate designation for the type of modem you have. Be sure your modem is set up properly in the /etc/acucap file; see acucap(5) and the comments in the /etc/acucap file for assistance.

2.4.9 uucpsetup Does Not Set Up tty Shared Mode Lines

A new feature of this release is the shared modem line, used for both incoming and outgoing dial-up calls (uucp or tip).

The uucpsetup command does not set up the shared modem lines. To take advantage of the shared modem lines, follow these steps:

1. Read the *ULTRIX-32 System Manager's Guide* for instructions on how to set up the /etc/ttys file and how to set up shared lines. You should also see ttys(5) in the *ULTRIX-32 Programmer's Manual* for more information.
2. Edit the /etc/ttys file and change the status line from off modem to on modem shared.
3. Type the following command to make the change effective:

   ```
   # kill -1 1
   ```

2.5 Known Problems and Corrections

This section describes known problems that may affect your ULTRIX-32 operating system once it has been installed and is running. The corrections to these problems are also described.

2.5.1 Halting a Running VAX 8600 or VAX 8650 Processor Can Cause Failures

If you halt an ULTRIX-32 system running on a VAX 8600 or VAX 8650 processor (by pressing CTRL/P), unpredictable failures may result. If you then issue a CONTINUE command at the console subsystem prompt (>>>), the system may crash.

These failures may require that you reboot the console subsystem. For example:

```
    >>> BOOT
```

2.5.2 Hardwired Terminals May Not Respond After Being Turned Off

Hardwired terminal ports other than the console port may hang as a result of electrical noise appearing on the line when the terminal is turned off and then back on. When the port is hung, the terminal does not respond to keyboard input. To temporarily reactivate the terminal line, follow these steps:

1. Determine the process identification (PID) of the getty process associated with the terminal line (you can use the ps(1) command).
2. Use the kill(1) command to kill that process. If you do not know the number of the hung terminal line, use the last(1) command with the user's log-in name.

The following example shows how to temporarily reactivate a terminal line, if the log-in name of the user of the hung terminal is anders:

```
# last anders
anders   ttyh3     Mon Nov 18 10:00   still logged in
anders   ttyh3     Mon Nov 18 08:35 - 09:08  (00:33)
anders   ttyh3     Mon Nov 18 00:26 - 02:00  (01:33)

# ps -ax

  PID TT STAT  TIME COMMAND
    0 ?  D     0:01 swapper
    1 ?  I     0:34 init
    2 ?  D     0:00 pagedaemon
  159 h3 I     0:00 - 2 ttyh3 (getty)
  160 Iq I     0:00 - 2 ttyIg (getty)

# kill -9 159
```

To permanently fix the terminal line, follow these steps:

1. Edit the /etc/ttys file. For each line that has a hardwired terminal (each terminal that does not use a modem), change T*nnnn* to std.*nnnn*, where *nnnn* is the speed of the terminal connected to the hardwired line.

   For example, assume this entry appears in the /etc/ttys file:

```
ttyA1  "/etc/getty T9600"  vt100   on nomodem secure  # direct connect
```

Change T9600 to std.9600:

```
ttyA1  "/etc/getty std.9600"  vt100   on nomodem secure  # direct connect
```

2. Type the following command to make the change effective immediately:

   ```
   # kill -1 1
   ```

2.5.3 Tape Drive May Hang on Remote System If Local System Crashes

The tape drive on the remote system may hang if the local system crashes while you are using the rdump(8c) or rrestore(8c) commands.

The rdump and rrestore commands request the remote system (with the tape drive) to start the /etc/rmt process, which then issues commands to the tape drive.

If the local system crashes, it loses contact with the remote /etc/rmt process and therefore can not terminate the process and release the tape drive. This causes the remote system to act as though the tape drive were no longer accessible by other processes on that system, because the remote system thinks the tape drive is still allocated to the /etc/rmt process.

Follow these steps to correct the problem:

1. On the remote system, type the following command to find the /etc/rmt process:

   ```
   # ps -alx | grep /etc/rmt
   ```

2. Find the process number for the /etc/rmt process from the output of the ps(1) command. In this example, the process number (PID) is 25:

```
  F UID   PID  PPID CP PRI NI ADDR   SZ   RSS WCHAN STAT TT   TIME COMMAND
801  0    25    24  2  -5  0 658e   18    14 830d0 D     ?    0:00 /etc/rmt
```

3. Kill the /etc/rmt process:

```
        # kill -9 25
```

4. Rewind the tape:

```
        # mt rew
```

2.5.4 instmedia Causes Error After Use

The procedure that the installation software uses to load the software subsets, instmedia(8), leaves the /usr partition mounted after you load all subsets. This causes an error to appear when you bring the system to multiuser mode.

After you finish installing the subsets you choose and before you bring up the system to multiuser mode, change your working directory to root and unmount the file systems using the -a option with the umount(8) command:

```
    # cd /
    # /etc/umount -a
```

2.5.5 netsetup Command May Ask for System Name

When installing your network with the netsetup(1) command at any time other than the initial installation, netsetup may incorrectly warn you that you have not given your system a name. If you have already set your host name in the /etc/rc.local file, disregard this message:

```
    Your system does not have a name.
    You must use the hostname command to name
    your system.
```

At the next prompt, type the name for your system:

```
    Enter the name for your system: myvax
```

If you have not yet named your system, add your system's name to the /etc/rc.local file.

2.5.6 uucpsetup Installs Modem Line on ttyh2

The uucpsetup command assigns modem lines to ttyh1 and ttyh2 if it finds a DMF32 controller. These lines are renamed as ttyd1 and ttyd2.

On the DMF32 controller, however, only the first two lines support modem control, so ttyh2 is assigned to a line without modem control.

To fix the problem, follow these steps:

1. Change your working directory to /dev, rename ttyh0 to ttyd2, and make a new special file for ttyh2:

```
# cd /dev
# mv ttyh0 ttyd2
# /etc/mknod ttyh2 c 22 2
```

2. In the /etc/ttys file, change ttyh0 to ttyh2. Here is how the modified /etc/ttys entries should look:

```
ttyh2  "/etc/getty T9600"  vt100  on nomodem   # direct connect
ttyd1  "/etc/getty D1200"  vt100  on modem     # dial-in port
ttyd2  "/etc/getty D1200"  vt100  off modem    # dial-out port
```

2.5.7 uucp and tip May Not Work with DMF32

If you have an autodial modem connected to a DMF32 interface and you are using the generic dialer routines in acucap(5), tip(1c) or uucp(1c) may not be able to open the modem and may print either of the following messages:

```
tip: can't synchronize
uucp: can't synchronize
```

The problem occurs because the DMF32 can not return characters to the system until after a carrier is detected by the modem.

If you encounter this problem, edit each entry in /etc/acucap that refers to a modem connected to a DMF32 by including the si boolean (see acucap(5) in the Programmer's Manual for more information). Including si boolean disables checking of responses from the modem until carrier is detected.

2.5.8 uucp May Not Work After Update From Version 1.1

uucp may not work after you have installed the Version 1.2 software update distribution kit because of a problem with links to terminal special files in /dev. If uucp does not work on your system, do the following:

1. Look in the /dev directory for files whose link count is greater than one:

```
# ls -li /dev
      .
      .
      .
 2072 crw--w--w-  3 root   1,  2 Feb  1 16:44 cua02
      .
      .
      .
 2120 crw--w--w-  1 root   1,  1 Jan 15 07:04 tty01
 2072 crw--w--w-  3 root   1,  2 Feb  1 16:44 tty02
      .
      .
      .
 2072 crw--w--w-  3 root   1,  2 Feb  1 16:44 ttyd2
      .
      .
      .
```

2. Compare the inode numbers for any files with link counts greater than one. For each group of files with multiple links, remove all the links except one:

   ```
   # rm cua02 ttyd2
   ```

3. Use the mknod(8) command to make new special files for each of the links you removed in Step 2, using the appropriate major and minor device numbers:

   ```
   # mknod /dev/cua02 c 1 2
   # mknod /dev/ ttyd2 c 1 2
   ```

2.5.9 Do Not Make Links to /dev/tty Devices

Do not make links to any /dev/tty device, including aliases to terminal names (such as ttyd2 linked to tty02 or ttyd1 linked to tty01).

Instead of making a link, create a node with the new name and the same major and minor device numbers. For example, if you require a link to tty02 with major and minor device numbers of 1 and 2, in that order, use the following command line:

```
# mknod /dev/ttyd2 c 1 2
```

2.5.10 /usr/adm/aculog Mode May Cause Security Problem

As distributed, the /usr/adm/aculog file has a mode of 666, which allows anyone to read it or write to it. This file may contain telephone numbers and other information that should not be public.

To make this file unreadable by regular system users, type:

```
# chmod 600 /usr/adm/aculog
```

2.5.11 C Optimizer Fails

The C optimizer does not process the following program segment correctly:

```
register char * from = 0;
if (*from ++ & 1)
```

The problem occurs when a pointer to a character is a register variable and the pointer is postincremented while testing the low bit of the character. The compiler generates the following code for the previous example (assuming from is stored in r11):

```
bitb     $1,(r11)+
beql     Label
```

The optimizer erroneously converts this to the following:

```
blbc     (r11)+,Label
```

The problem occurs because the blbc has a longword source operand while the bitb instruction has a byte operand. Thus the optimized case incorrectly increments the pointer by 4 rather than by 1.

To avoid this problem, do one of the following:

- Do not store the pointer in a register when the low bit of the character will be tested in the same operation as the pointer is being incremented.
- Increment the pointer after the bit has been tested. For example:

```
        if (*from & 1)
                from++;
```

- Remove the routine that contains the problem segment and store it in another file. Then compile that file without the optimizer flag.

  By storing the routing containing the problem segment in a separate file, you can use the optimizer flag with the rest of the program.

2.5.12 f77 Fails To Resolve All External Symbols

The f77 compiler fails to resolve all external symbols when linking a Fortran program that contains a call to ioinit(3f). For example:

```
    call ioinit(.true.,.false.,.false.,FORT,.false.)
         stop
         end

    #f77 example.f
```

In this example, the error messages are:

```
    example.f:
    MAIN:
    Undefined:
    _i_len
    _i_indx
    _lnblnk_
    _s_cmp
    _s_copy
```

To avoid this problem, include -lI77 in the command line that invokes the compiler. For example:

```
    # f77 example.f -lI77
```

2.5.13 Entry in /etc/acucap Does Not Work for Ventel Modems

The entry for Ventel modems in the /etc/acucap file does not work on shared terminal lines, because they always assume that carrier is present.

To enable a Ventel modem to work on a nonshared terminal line, replace its entry in the /etc/acucap file with the following:

```
ventel|ventel md212-3e:\
    :hu:ss=\r\r:sr=VEN-TEL:sd#1:di=K:dt=\r:\
    :dr=DIAL:fd#30:dd#1:da#5:
```

2.5.14 Interaction Between DECnet and arp Causes Network Problem

When DECnet begins running on a node, it changes the default Ethernet address. Other ULTRIX systems on the same Ethernet that have connected to that node before DECnet started will then have the wrong address for your node, and attempts to connect with it will time out.

To temporarily correct this problem: After DECnet is running on your node, delete the incorrect address entry on each remote system using the arp(8) command. The next time one of the remote nodes attempts to contact your node, the correct Ethernet address will be determined and stored.

For example, if your node's name is emily, log in to each of the remote systems as root and enter the following command:

```
# /etc/arp -d emily
```

To fix the problem permanently: Be sure the ncp entry comes before the /etc/ifconfig entry in the /etc/rc.local file. This ensures that DECnet is started before the local network configuration is established and connections can be made from remote ULTRIX systems.

The following example shows how the first seven lines of the /etc/rc.local file should look:

```
#
# @(#)rc.local
/bin/hostname emily
/etc/convert_nodedb
/usr/bin/ncp set executor state on ; echo 'starting DECnet.' >/dev/console
/etc/ifconfig de0 `/bin/hostname` broadcast 98.0 netmask 255.255.255.0
/etc/ifconfig lo0 localhost
```

2.5.15 script Command Interprets CTRL/Z and CTRL/C Signals

The script(1) command does not handle keyboard-generated signals properly. The script command creates a file containing everything printed on your terminal. It does this by forking a new instance of the shell and capturing all terminal output received.

Normally, signals generated by the keyboard are propagated to the new shell and handled properly. In this release, however, the script command receives the signals instead of passing them on. As a result, typing

CTRL/C causes script to terminate prematurely and leave the terminal in a state where no characters echo. If this happens, you can reset your terminal by typing:

```
    # reset
```

Remember that no characters are echoed as you type the reset command.

2.5.16 Improper Use of sccs fix Command Leaves File Unusable

The sccs fix command replaces a delta already merged into an SCCS s-file. If you inadvertently use the fix command on an s-file that you have just created with the sccs admin command, the initial delta is removed and the s-file is placed in an unusable state. This is an illegal use of the fix command, but SCCS does not check for it. SCCS removes the delta and leaves the s-file without a valid SCCS version number (SID).

Once the s-file is in this state, you can not use SCCS commands to make changes or extract copies of the file. To correct the problem, follow these steps:

1. Become superuser.
2. Delete the s-file.
3. Create a new s-file using the sccs admin command.

2.5.17 beautify Variable in tip Can Not Be Unset

You can not unset the beautify variable in the tip(1) utility. By default, the beautify variable is set.

2.5.18 fortunes.dat File Has Wrong Permission Modes Set

The fortunes.dat file has the wrong permission modes set, and thus the fortune(6) command does not work.

Change the mode of /usr/games/lib/fortunes.dat to readable and executable for all groups:

```
    # chmod 555 /usr/games/lib/fortunes.dat
```

2.5.19 On-Line Documentation for mtio is Incomplete

The on-line documentation for the mtio(4) special file is incomplete. The *ULTRIX-32 Programmer's Manual* includes an updated description of mtio(4).

2.6 General Notes

This section contains informational notes about changes made to this release of which you should be aware.

2.6.1 Capacity Upgrade Kits from Version 1.1 Do Not Work

The Capacity Upgrade kits you may have purchased for use with ULTRIX-32 Version 1.1 will not work with Version 1.2.

If you have a Software Services contract, you should receive a replacement Capacity Upgrade kit.

If you do not have a Software Services contract but you need to increase the maximum number of users on your system, you must purchase the distribution kit for the Capacity Upgrade package that corresponds with your system. Contact your DIGITAL representative for more information.

2.6.2 New Format for /etc/ttys File

The format of the /etc/ttys file is changed in this release. The /etc/ttys files from earlier versions of ULTRIX-32 software are not compatible with this release, and the /etc/ttytype file is no longer used. See ttys(5) in the Programmer's Manual for more information.

2.6.3 Some User Programs Must Be Recompiled

If you have user-supplied programs that reference the kernel data structures through /dev/mem, you must recompile them before using them with the ULTRIX-32 Version 1.2 system.

2.6.4 Default Filter Needed in Printcap File for LP11 Printers

The LP11 driver supplied in this release does not filter the data stream sent to it; previous versions did not require a filter if the LP11 was connected to a dumb line printer.

This change requires that your system's /etc/printcap file contain a line specifying /usr/lib/lpf as the default output filter for printers connected to LP11 controllers. If you do not specify a default filter, data sent to the printer is not formatted with tabs and page length unless the data has already been filtered by the pr(1) command.

To specify /usr/lib/lpf as the default filter for a dumb printer connected to an LP11 controller, add the following to the appropriate entry in /etc/printcap:

```
    :of=/usr/lib/lpf:
```

2.6.5 Spool Directories for Remote Printers

If you specify a printcap entry for any printer, you must also create a spool directory for that printer. For example, if you have an LA100 printer, type this command to create a spool directory for it:

```
    # mkdir /usr/spool/la100
```

2.6.6 SO_DONTLINGER Option No Longer Exists.

The socket option SO_DONTLINGER no longer exists. The linger structure is a new option to setsockopt(2) that allows you to turn the linger structure on or off. See socket(2) in the Programmer's Manual for further information.

2.6.7 SO_BROADCAST Option Added

A new socket option called SO_BROADCAST has been added. The SO_BROADCAST option must be set by setsockopt(2) to send a broadcast message. See socket(2) in the Programmer's Manual for further information.

2.6.8 dskx May Corrupt Disk or Destroy Superblock

If not used properly, the dskx(8) command could corrupt the disk being tested or destroy the disk's superblock.

The dskx exerciser with the -p and -c options reads and writes random data to test disks. The -p option overwrites any data on the file system being tested. The -c option overwrites any data on the entire device (disk) being tested. If the dskx exerciser discovers a superblock on the partition being tested, it will not overwrite that partition unless you tell it to do so. Be careful, however, because many partitions overlap others. If the partition you are testing happens to overlap another partition with a superblock, the data on the overlapped partition will be destroyed.

For example, the h partition on an RA81 disk overlaps the d, e, and f partitions. If you are using the h partition, you can not test the d, e, or f partitions without corrupting the h partition.

For another example, the g partition of an RM05 disk overlaps the d, e, and f partitions. If you are using the g partition, you can not test the d, e, or f partitions without corrupting the g partition.

See Chapter 7, Disk Allocation, and Appendix C, Disk Partitioning, in the *ULTRIX-32 System Manager's Guide* for further information about disk partitions.

Configuring the MICOM PAD for tip and uucp A

This appendix provides information on how to configure the MICOM[1] Micro800/X.25 Packet Assembler Disassembler (PAD) for tip(1) and uucp(1) ULTRIX-32 software utilities. The MICOM Micro800/X.25 PAD hardware is not sold or supported by Digital Equipment Corporation.

A.1 Description of the MICOM PAD

The MICOM Micro800/X.25 PAD is sold and supported by MICOM Systems Inc. The PAD implements the Consulting Committee for International Telephony and Telegraphy (CCITT) recommendations X.3, X.28, and X.29. It connects directly to an X.25 access line (also called a trunk) and performs the necessary operations to place calls and create connections to other data terminal equipment (DTE) on the X.25 network. Asynchronous devices such as terminals or multiplexer lines gain access to the PAD by being connected by a cable to one of its channels.

Depending on the model purchased, the PAD has from 4 to 16 channels for sending or receiving X.25 calls. By attaching the PAD channels to terminal multiplexer lines on a processor running ULTRIX-32 software, you can configure the PAD to support outgoing uucp connections, outgoing tip connections, and incoming uucp and login services.

Although each channel can be configured for more than one purpose, in this discussion assume each channel is configured for a specific purpose. For example, channel 1 is for outgoing uucp calls, channel 2 is for outgoing tip calls, channel 3 is for incoming uucp connections, and channel 4 is for incoming log-in service. This implies that shared lines are not supported.

A.2 Connecting the PAD

Before connecting multiplexer lines to PAD channels, follow these steps:

1. Decide which channels to dedicate for which purposes, and then reserve that many multiplexer lines on your ULTRIX-32 system. These lines must support modem control.

[1] MICOM is a trademark of MICOM Systems, Inc.

2. Turn off any getty processes that may be running on the dedicated multiplexer lines:
   a. Edit the /etc/ttys file to make sure that the status of the dedicated lines is off.
   b. Send a HUP signal to the init process by typing:

      ```
      # kill -HUP 1
      ```

      The HUP signal makes the changes in the /etc/ttys file take effect.

The multiplexer lines can now be connected to the PAD channels using DIGITAL RS-232 modem-compatible cables.

Before setting up the PAD, read the *MICOM Micro800/X.25 Concentrator PAD Users Manual* to become familiar with the device. The PAD documentation contains the necessary information for configuring and using the PAD. If you are setting up the PAD for the uucp or tip utilities, first acquaint yourself with the PAD by using it with a terminal until you are able to successfully place calls and create connections to remote devices. Then review the sections in the *Micro800/X.25 Concentrator PAD User's Manual* for information on the specific configuration parameters to make the PAD work with uucp, tip, and remote login.

The PAD makes provisions for placing calls between its channels as a form of loopback. With a loopback, calls can be made without ever going over the X.25 network.

For example, suppose you have connected terminals to channels 3 and 4. To place a call from channel 3 to channel 4, type the following at channel 3:

```
C 00/04
```

The DTE address 00 places calls between channels.

A.3 Setting Up uucp for the PAD

This section describes how to set up uucp for use with the PAD. The uucp facility now has an additional protocol called *f-proto*, which is much faster than the standard g-proto. The calling machine initiates the f-proto protocol; however, both machines must recognize the protocol for it to be effective. See Section A.3.2 for information on how the ULTRIX-32 software can initiate the f-proto protocol. Furthermore, the PAD parameters shown in the profile below must be set for both the PAD channel placing the call and the PAD channel receiving the call.

A device profile is a list of PAD parameter values. When a device profile is assigned to a channel, the channel's PAD parameters take on the values listed in the device profile. You can define up to eight device profiles, numbered from 1 to 8. Reserve a device profile number for uucp and set the profile as shown below. For the following example, assume profile 8 was chosen for uucp.

You should assign the same device profile to all channels dedicated for uucp, for example, profile 8. Profile 8 needs the following PAD parameters set:

```
x28acc=0         bits/char=3
echo=0           dv_parity=0
data_fwd=2       net_parity=0
idletimer=10     c.xon=17
dv_flow=1        c.xoff=19
s.signal=0       special_flow=0
break=0          count_fwd=0
cr_pad=0         escdelay=0
l.fold=0         c.break=0
speed=14         c.supp=0
p.flow=1         c.subs=0
autolf=0         echoctrl=0
lf_pad=0         special_echo_id=0
edit=0           pagelength=0
c.del=0          c.page=0
l.del=0          ff_pad=0
l.disp=0         inactivity=0
dv.type=2        x29acc=0
echomask=0
```

Note that the speed parameter selected is 9600 baud (value 14). Refer to the MICOM documentation for information on how to change the speed parameter.

A.3.1 Setting Up Incoming uucp Lines

This section describes how to set up incoming uucp lines.

Configure the channel dedicated for incoming uucp calls as follows:

```
Profile: 8
Call option: 1
Address id Mnemonic: **
Channel Option: 6
```

Follow these steps:

1. Set the profile selection to the number assigned previously. The number 8 is used in this example because the profile was assigned 8 in the previous section. Be sure the channel option is 6. Channel option 6 raises the carrier detect signal each time a call comes in. Be sure the carrier detect signal is passed through the cable to the multiplexer line. The situation is analogous to having a dial-up modem installed. Leave the other factory-set values alone.
2. Place a getty process on the line. To do this, edit the /etc/ttys file and change the status of the line to on modem. Then, send a hangup signal to init by typing:

   ```
   # kill -HUP 1
   ```

   Assume line ttyh6 is connected to the PAD channel 3 and is reserved for uucp incoming connections. The following is an example entry for the /etc/ttys file:

   ```
   ttyh6 "/etc/getty T9600" vt100  on  modem  # PAD chan 3
   ```

   The result is a PAD channel that can be called over the X.25 network by another machine running uucp to request the f-proto.
3. Configure the channel reserved for outgoing uucp connections as follows:

   ```
   Profile: 8
   Call option: 1
   Address id Mnemonic: **
   Channel Option: 0
   ```

The channel option 0 means that the channel is dedicated.

A.3.2 Setting Up uucp to Dial Out Over the PAD

After you have configured the PAD channels for uucp dialout, you are ready to set up the uucp utility. For information on how to set up uucp, see the *ULTRIX-32 System Manager's Guide*. Then follow these steps:

1. Edit the /usr/lib/uucp/L-devices file to reflect the addition of the PAD dial-out lines. Here is the new format for the L-devices file:

type line call-unit speed brand preferred_proto

The PAD channels are treated as direct lines. The L-devices file now contains the *preferred protocol* field from which you request f-proto for lines connected to PAD dial-out channels. For example, suppose the PAD channel for uucp dialout is connected to ttyh5. The proper L-devices entry is:

```
DIR ttyh5 ttyh5 9600 direct f
```

For example, suppose you have two channels configured for uucp dialout: one connected to ttyh7, and the other to ttyh8. Here are the proper entries:

```
DIR ttyh7 ttyh7 9600 direct f
DIR ttyh8 ttyh8 9600 direct f
```

2. Edit the L.sys file. This file holds entries to call other machines. It contains the information necessary to call the host and perform the log-in sequence. To call a host over the X.25 network, assume that it is on a direct-connect line attached to the dial-out channel of the modem, and then include the PAD commands to call the host as part of the log-in sequence. For example, suppose the name of the node you are calling is mozart, the X.25 DTE address is 12345, and the PAD channel you are calling on mozart (the one dedicated for uucp login) is channel 3. Your PAD uucp dial-out channel is connected to ttyh5, the log-in name is Umozart, and the password is donny. The following L.sys entry could be used to call node mozart:

```
mozart Any DIR 9600 ttyh5 "" \r "" C\s12345/03 ogin:-\r-ogin: Umozart ssword: donny
```

In the above entry, the information that places the call over the PAD is C\s12345/03 and is interpreted as follows:

C	The PAD command to place an X.25 call.
\s	Indicates how to include a space character in an L.sys entry. Make sure you have a space between the PAD command (C) and the DTE address (12345).
12345	The DTE address.
/03	Subaddress channel 3 on the called PAD.

The rest of the line is the log-in sequence. Note that the device containing the PAD dial-out line is wired into the L.sys entry for calling node mozart. Thus, even if two PAD channels are reserved for uucp dialout, the same one is always used to call mozart. Rotary action is not possible.

A.4 Setting Up tip for Use with the PAD

This section describes how to set up the tip(1) utility for use with the PAD.

A.4.1 Setting Up Outgoing tip Connections

The PAD tip dial-out lines are treated like direct connects. For example, suppose you are on the X.25 network called telenet and your outgoing tip channel is connected to /dev/ttyh7. The proper entry for the /etc/remote file is:

```
    telenet|pad|PAD outgoing channel:\
            dv=/dev/ttyh7:br#9600
```

By typing the following, you would get connected to the PAD:

```
    # tip telenet
```

You can then place your X.25 call just as if you were connected directly to the PAD (you are in effect connected directly to the PAD).

If you have three outgoing tip channels connected to /dev/ttyh4, /dev/ttyh5, and /dev/ttyh6, then you should have this entry in the /etc/remote file:

```
    telenet|pad|PAD outgoing channel:\
            dv=/dev/ttyh4,/dev/ttyh5,/dev/ttyh6:br#9600
```

This causes a rotary effect. If the first channel is busy, tip tries connecting to the next channel, and so on.

When you have completed your call, remember to type ~. (tilde-dot) to make tip hang up the connection. The tip command has no other way of knowing when your call is over.

The PAD parameters specify information such as whether the PAD echoes characters and when the host PAD forwards characters it receives to the remote PAD. The standard ULTRIX-32 convention for tip(1) is that the remote host echoes characters. This convention is necessary for screen-oriented programs such as vi(1). However, it produces a high X.25 traffic rate (up to two X.25 packets for each character typed), and accordingly causes high public data network (PDN) charges. The other extreme is to have the PAD perform all the echoing and editing features, and then only forward characters when, for example, an entire line has been typed. Many programs do not function properly if they have to wait for an entire line before they can receive any characters.

There is no solution that fits all situations. However, if you use the following device profile, which causes high network charges, you do not lose any ULTRIX-32 software functionality:

```
x28acc=0          echomask=0
echo=0            bits/char=3
data_fwd=127      dv_parity=0
idletimer=10      net_parity=0
dv_flow=1         c.xon=17
s.signal=5        c.xoff=19
break=0           special_flow=0
cr_pad=0          count_fwd=0
l.fold=0          escdelay=0
speed=14          c.break=0
p.flow=1          c.supp=0
autolf=0          c.subs=0
lf_pad=0          echoctrl=0
edit=0            special_echo_id=0
c.del=0           pagelength=0
l.del=0           c.page=0
l.disp=0          ff_pad=0
dv.type=2         inactivity=0
                  x29acc=0
```

For purposes of this example, assume profile 5 has been assigned the values shown above. Profile 5 forwards X.25 packets for every character typed. Also, although the PAD displays the PAD service prompt (s.signal=5), it does not echo the characters typed as you place the call because of the echo=0 parameter.

Remember, this profile (profile 5) may not be proper for your installation. The PAD parameter information in your PAD documentation gives other examples where the PAD is used to echo the characters and perform editing.

Configure the channels reserved for outgoing tip connections as follows:

```
Profile: 5
Call option: 1
Address id Mnemonic: **
Channel Option: 0
```

Note that the parameter for Profile is the only variable. The other parameters must have the values shown.

A.4.2 Setting Up Incoming login Service

The same issues related to outgoing tip connections are applicable with incoming login service. Follow these steps:

1. Set up the profile. Use the same profile for incoming login services that you use for outgoing tip connections. Configure the channels dedicated for incoming login calls as follows:

   ```
   Profile: 5
   Call option: 1
   Address id Mnemonic: **
   Channel Option: 6
   ```

   The important setting is the channel option. Channel option 6 specifies that the channel will raise carrier detect when a call comes in. Be sure that the carrier detect signal is passed through the cable to the multiplexer line. The situation is analogous to having a dial-up modem installed.

2. Place a getty process on the line. First, edit the /etc/ttys file and change the status of the line to be on modem. Then, send a HUP signal to init by typing:

   ```
   # kill -HUP 1
   ```

Assuming line ttyh6 is connected to the PAD channel 3 and is reserved for incoming login connections, the following is an example entry for the /etc/ttys file:

```
ttyh6   "/etc/getty T9600" vt100  on  modem  # PAD chan 3
```

When a call comes over the X.25 network for a channel designated for incoming login service, the channel raises the carrier detect signal, the getty waiting on the line wakes up, and a log-in message appears for the user placing the call.

New and Replacement Programmer's Manual Pages B

This appendix contains updated and new pages for the *ULTRIX-32 Programmer's Manual.* Some of these pages are new versions of existing pages; others are completely new pages to be inserted among the existing pages.

Instructions

Replace the following pages in the *ULTRIX-32 Programmer's Manual, Sections 1 and 6:*

Old Page	New/Replacement Page
1-247/1-248	1-247/1-248
1-249/1-250	1-249/1-250
–	1-250a/blank
1-383/1-384	1-383/1-384
1-385/1-386	1-385/1-386

Add the following pages to the *ULTRIX-32 Programmer's Manual, Sections 2 and 3:*

Old Page	New/Replacement Page
–	3-226a/3-220b
–	3-226c/blank

Insert or replace the following pages in the *ULTRIX-32 Programmer's Manual, Sections 4, 5, 7, and 8:*

Old Page	New/Replacement Page
4-17/4-18	4-17/4-17a
–	4-18/blank
4-73/4-74	4-73/4-73a
–	4-73b/4-74
5-49/5-50	5-49/5-50

NAME

man − find manual information by keywords; print out the manual

SYNTAX

man −**k** keyword ...
man −**f** file ...
man [−] [−**t**] [section] title ...

DESCRIPTION

Man is a program which gives information from the programmers manual. It can be asked for one line descriptions of commands specified by name, or for all commands whose description contains any of a set of keywords. It can also provide on-line access to the sections of the printed manual.

When given the option −**k** and a set of keywords, *man* prints out a one line synopsis of each manual sections whose listing in the table of contents contains that keyword.

When given the option −**f** and a list of file names, *man* attempts to locate manual sections related to those files, printing out the table of contents lines for those sections.

When neither −**k** nor −**f** is specified, *man* formats a specified set of manual pages. If a section specifier is given *man* looks in the that section of the manual for the given *titles*. *Section* is an Arabic section number (3 for instance). The number may followed by a single letter classifier (1g for instance) indicating a graphics program in section 1. If *section* is omitted, *man* searches all sections of the manual, giving preference to commands over subroutines in system libraries, and printing the first section it finds, if any.

If the standard output is a teletype, or if the flag − is given, *man* pipes its output through *cat*(1) with the option −s to crush out useless blank lines, *ul*(1) to create proper underlines for different terminals, and through *more*(1) to stop after each page on the screen. Hit a space to continue, a control-D to scroll 11 more lines when the output stops.

The −t flag causes *man* to arrange for the specified section to be *troff'ed* to a suitable raster output device; see *vtroff*(1).

FILES

/usr/man/man?/*
/usr/man/cat?/*

SEE ALSO

more(1), ul(1), whereis(1), catman(8)

RESTRICTIONS

The manual is supposed to be reproducible either on the phototypesetter or on a typewriter. However, on a typewriter some information is necessarily lost.

mdtar(1)

NAME

mdtar – multiple diskette archiver

SYNTAX

mdtar [–] [*key*] [*name* ...]

DESCRIPTION

The *mdtar* command saves multiple files on multiple archives (usually an RX50 diskette, but any file or device may be specified). The *mdtar* command supports special files as well as regular files.

A dash (–) specified before the *key* argument is not necessary and makes no difference to *mdtar*'s performance. The *mdtar* command's actions are controlled by the *key* argument. The *key* is a string of characters containing one function letter and one or more function modifiers. Other arguments to *mdtar* are file or directory names specifying which files to dump or restore. In all cases, appearance of a directory name refers to the files and, recursively, subdirectories of that directory. The *mdtar* command also saves special files. The default device used by *mdtar* is */dev/rra1a*.

The function portion of the *key* is specified by one of the following letters:

c Create a new archive. Writing begins on the beginning of the archive instead of after the last file. This command implies **r**.

r Write the named files to the end of the archive. The **c** function implies this.

t List the names of the files as they occur on the input device.

u Update the current archive. Add the named files to the archive, if they are not there already or if they have been modified since they were last put on the archive.

x Extract the named files from the archive. If the named file matches a directory whose contents had been written onto the archive, this directory is recursively extracted. The owner, modification time, and mode are restored if you are the superuser and if you have specified the **p** function. If no file argument is given, the entire content of the archive is extracted. If multiple entries specifying the same file are on the archive, the last one overwrites all previous versions extracted.

You can use one or more of the following function modifiers in addition to the letter which selects the function desired:

A Use the next argument as the archive number to begin output. This function modifier is intended for error recovery. The *mdtar* command outputs files in terms of archives. Each archive contains a number of files. For example, if *mdtar* has been requested to dump a path (set of files) that consists of 10 archives and there is an error writing the *n*th archive, then you can specify the **A** function modifier to restart *mdtar* at the *n*th archive.

Caution

You must issue the same path (set of files) as in the first command. This guarantees that *mdtar* will begin at the correct file on archive *n.*

If the v function modifier is specified, *mdtar* outputs informational messages to inform you of its progress. For example, the following command dumps the entire directory structure:

```
# mdtar cv /
```

If an error occurs on archive seven, to restart at the seventh archive. without having to dump the first six archives again, type:

```
# mdtar cvA 7
```

The *mdtar* command will tell you it is skipping the first six archives and that it will resume output with the data that begins at archive seven.

B Force input and output blocking to 20 blocks per record. This function modifier allows *mdtar* to work across a communications channel where the blocking may not be maintained.

F[F] Operate in *fast mode*. When −F is specified, *mdtar* skips all SCCS directories, core files, and errs files. When −FF is specified, *mdtar* also skips all *a.out* and **.o* files.

H Help mode. Print a summary of the keys and function modifiers.

V Display extended verbose information. Included are the version number of *mdtar*, the number of blocks used on the device, the number of blocks in a file, and the protection modes given in a format similar to the ls −l command. In addition to this information, V provides the information given by the v function modifier.

b Use the next argument as the blocking factor for archive records. The default is 20 (the maximum).

f Use the next argument as the name of the archive instead of *dev/rra1a*. If the name of the file is −, *mdtar* writes to standard output.

g Use */dev/rgt0* as the default device.

h Save a copy of the actual file on the output device under the symbolic link name, instead of placing the symbolic information on the output. The default action of *mdtar* is to place symbolic link information on the output device. A copy of the file itself is not saved on the output device.

k	Use /dev/rtk0 as the default device, instead of */dev/rra1a*.
i	Ignore checksum errors found in the archive.
l	Print an error message if all links to the files dumped can not be resolved. If –l is not specified. no error messages are printed.
m	Do not restore the modification times. The modification time is the time of extraction.
n	Select /dev/rmt0 as the default device.
o	Suppress the normal directory information. On output, *mdtar* normally places information specifying owner and modes of directories in the archive. Former versions of *tar*, when encountering this information gave error messages of the form: `<name>/: cannot create`
p	Restore the named files to their original modes, ignoring the present *umask*(2). Setuid and sticky bit information is also restored to the superuser. (See *stat*(2), and S_ISVTX).
s	The next argument specifies the size of the output archive media in 512 byte blocks. This enables *mdtar*() to be used with devices of different physical media sizes. The default is 800 blocks (assumption is an RX50 output archive).
v	Write the name of each file treated, preceded by the function letter, to diagnostic output. Normally *mdtar* does its work silently. With the t function modifier, the verbose function gives more information about the archive entries than just their names.
w	Print the action to be taken followed by file name, then wait for user confirmation. If a word beginning with the letter *y* is given, the action is done. Any other input means do not do it.
0...9	Select the named drive as an alternate drive on which the archive is mounted. The default is */dev/rra1a*.

If a file name is preceded by –C. then *mdtar* performs a *chdir*(2) to that file name. This allows multiple directories not related by a close common parent, to be archived using short relative path names. For example, to archive files from the */usr/include* and */etc* directories, type:

```
# tar c -C /usr/include -C /etc
```

FILES

/dev/rmt8
/dev/rmt0
/dev/rtk0
/dev/rgt0
/dev/rra1a
*/tmp/mdtar**

RESTRICTIONS

The u option can be slow.
The current limit on file name length is 100 characters.
There is no way to follow symbolic links selectively.

DIAGNOSTICS

Indicates bad key characters and archive read/write errors.
Indicates if enough memory is not available to hold the link tables.

SEE ALSO

stat(2), *tar*(5)

NAME

tar – tape archiver

SYNTAX

tar [–] [*key*] [*name* ...]

DESCRIPTION

The tape archiver command, *tar*(1), saves and restores multiple files on a single file (usually a magnetic tape, but it can be any file). The *tar* command supports special files as well as regular files.

A dash (–) specified before the *key* argument is not necessary and makes no difference to *tar*'s performance. The *tar* command's actions are controlled by the *key* argument. The *key* is a string of characters containing at most one function letter and possibly one or more function modifiers. Other arguments to *tar* are file or directory names specifying which files to dump or restore. In all cases, the appearance of a directory name refers to the files and (recursively) subdirectories of that directory. The default device used by *tar* is */dev/rmt8.*

The function portion of the *key* is specified by one of the following letters:

c Create a new tape. Writing begins on the beginning of the tape instead of after the last file. This command implies **r**.

r Write the named files to the end of the tape. The **c** function implies this.

t List the names of the files as they occur on the input device.

u Add the named files to the tape if they are not there already or if they have been modified since they were last put on the tape.

x Extract the named files from the tape. If the named file matches a directory whose contents had been written onto the tape, this directory is recursively extracted. The owner, modification time, and mode are restored, if possible. If no file argument is given, the entire content of the tape is extracted. Note that if multiple entries specifying the same file are on the tape, the last one overwrites all previous versions extracted.

You can use one or more of the following function modifiers in addition to the letter which selects the function desired:

B Force input and output blocking to 20 blocks/record. This function modifier allows *tar* to work across a communications channel where the blocking may not be maintained.

F[F] Operate in *fast mode*. When **–F** is specified, *tar* skips all SCCS directories, core files, and error files. When **–FF** is specified, *tar* also skips all *a.out* and **.o* files.

H Help mode. Print a summary of the keys and function modifiers.

V Display extended verbose information. Included are the version number of *tar*, the number of blocks used on the device, the number of blocks in a file, and the protection modes given in a format similar to the *ls −l* command. In addition to this information, V provides the information given by the v function modifier.

b Use the next argument as the blocking factor for tape records. The default is 20 (the maximum). This option should only be used with raw magnetic tape archives (See the f function modifier). The block size is determined automatically when reading tapes (key letters x and t).

d Use */dev/rra1a* as the default device (blocking factor of 10).

f Use the next argument as the name of the archive instead of */dev/rmt8*. If the name of the file is −, *tar* writes to standard output or reads from standard input, whichever is appropriate. Thus, *tar* can be used as the head or tail of a filter chain. You can also use *tar* to move hierarchies. The following example shows how to move the directory *fromdir* to the directory *todir*:

```
    # cd fromdir; tar cf - . | (cd todir; tar xf -)
```

g Use */dev/rgt0* as the default device, instead of */dev/rmt8*.

h Save a copy of the actual file on the output device under the symbolic link name, instead of placing the symbolic information on the output. The default action of *tar* is to place symbolic link information on the output device. A copy of the file itself is not saved on the output device.

i Ignore checksum errors found in the archive.

k Use */dev/rtk0* as the default device, instead of */dev/rmt8*.

l Complain if *tar* cannot resolve all of the links to the files dumped. If this is not specified, no error messages are printed.

m Do not restore the modification times. The modification time will be the time of extraction.

n Select */dev/rmt0* as the default device, instead of */dev/rmt8*.

o Suppress the normal directory information. On output, *tar* normally places information specifying owner and modes of directories in the archive. Former versions of *tar*, when encountering this information provided error messages of the form:

```
<name>/: cannot create
```

p Restore the named files to their original modes, ignoring the present *umask*(2). Setuid and sticky bit information is also restored to the superuser.

v Write the name of each file treated, preceded by the function letter, to diagnostic output. Normally, *tar* does its work silently. With the **t** character, the verbose function modifier provides more information about the tape entries than just their names.

w Print the action to be taken, followed by file name, then wait for user confirmation. If a word beginning with the letter *y* is given, the action is done. Any other input means do not do it.

0...9 Select the named drive as an alternate drive on which the tape is mounted. The default is drive 0 at 1600 bpi, which is usually */dev/rmt8*.

If a file name is preceded by **–C**, then *tar* performs a *chdir*(2) to that file name. This allows multiple directories not related by a close common parent to be archived using short relative path names. For example, to archive files from */usr/include* and */etc*, type:

```
# tar c -C /usr/include -C /etc
```

The *tar* command can properly handle blocked archives.

FILES

/dev/rgt0
/dev/rmt0
/dev/rmt8
/dev/rtk0
/dev/rra1a
*/tmp/tar**

RESTRICTIONS

There is no way to ask for the *nth* occurrence of a file.
Tape errors are handled ungracefully.
The **u** key can be slow.
The current limit on file name length is 100 characters.
There is no way to follow symbolic links selectively.

DIAGNOSTICS

Indicates bad key characters and tape read/write errors.
Indicates if enough memory is not available to hold the link tables.

SEE ALSO

mdtar(1,) *tar*(5)

tbl(1)

NAME

tbl – format tables for nroff or troff

SYNTAX

tbl [files] ...

DESCRIPTION

Tbl is a preprocessor for formatting tables for *nroff* or *troff*(1). The input files are copied to the standard output, except for lines between .TS and .TE command lines, which are assumed to describe tables and are reformatted. Details are given in the *tbl*(1) reference manual.

EXAMPLE

As an example, letting \t represent a tab (which should be typed as a genuine tab) the input

```
.TS
c s s
c c s
c c c
l n n.
Household Population
Town\tHouseholds
\tNumber\tSize
Bedminster\t789\t3.26
Bernards Twp.\t3087\t3.74
Bernardsville\t2018\t3.30
Bound Brook\t3425\t3.04
Branchburg\t1644\t3.49
Bridgewater\t7897\t3.81
Far Hills\t240\t3.19
.TE
```

yields

```
    Household Population
      Town         Households
                 Number   Size
Bedminster          789   3.26
Bernards Twp.      3087   3.74
Bernardsville      2018   3.30
Bound Brook        3425   3.04
Branchburg         1644   3.49
Bridgewater        7897   3.81
Far Hills           240   3.19
```

NAME

malloc, free, realloc, calloc, mallopt, mallinfo – fast main memory allocator

SYNOPSIS

#include <malloc.h>
char *malloc (size)
unsigned size;

void free (ptr)
char *ptr;

char *realloc (ptr, size)
char *ptr;
unsigned size;

char *calloc (nelem, elsize)
unsigned nelem, elsize;

int mallopt (cmd, value)
int cmd, value;

struct mallinfo mallinfo (max)
int max;

DESCRIPTION

malloc and *free* provide a simple general-purpose memory allocation package, which runs considerably faster than the *malloc*(3) package. It is found in the library *malloc*, and is loaded if the option *–lmalloc* is used with *cc*(1) or *ld*(1).

Malloc returns a pointer to a block of at least *size* bytes suitably aligned for any use.

The argument to *free* is a pointer to a block previously allocated by *malloc*; after *free* is performed, this space is made available for further allocation, and its contents have been destroyed (see *mallopt* below for a way to change this behavior).

Undefined results will occur if the space assigned by *malloc* is overrun or if some random number is handed to, *free*.

Realloc changes the size of the block pointed to by *ptr* to *size* bytes and returns a pointer to the (possibly moved) block. The contents will be unchanged up to the lesser of the new and old sizes.

Calloc allocates space for an array of *nelem* elements of size *elsize*. The space is initialized to zeros.

Mallopt provides for control over the allocation algorithm. The available values for *cmd* are:

M_MXFAST Set *maxfast* to *value*. The algorithm allocates all blocks below the size of *maxfast* in large groups and then doles them out very quickly. The default value for *maxfast* is 0.

M_NLBLKS	Set *numlblks* to *value*. The above mentioned large groups each contain *numlblks* blocks. *Numlblks* must be greater than 0. The default value for *numlblks* is 100.
M_GRAIN	Set *grain* to *value*. The sizes of all blocks smaller than *maxfast* are considered to be rounded up to the nearest multiple of *grain*. *Grain* must be greater than 0. The default value of *grain* is the smallest number of bytes which will allow alignment of any data type. Value will be rounded up to a multiple of the default when *grain* is set.
M_KEEP	Preserve data in a freed block until the next *malloc*, *realloc*, or *calloc*. This option is provided only for compatibility with the old version of *malloc* and is not recommended.

These values are defined in the <malloc.h> header file.

Mallopt may be called repeatedly, but may not be called after the first small block is allocated.

Mallinfo provides information describing space usage. It returns the following structure:

```
struct mallinfo  {
        int arena;        /* total space in arena */
        int ordblks;      /* number of ordinary blocks */
        int smblks;       /* number of small blocks */
        int hblkhd;       /* space in holding block headers */
        int hblks;        /* number of holding blocks */
        int usmblks;      /* space in small blocks in use */
        int fsmblks;      /* space in free small blocks */
        int uordblks;     /* space in ordinary blocks in use */
        int fordblks;     /* space in free ordinary blocks */
        int keepcost;     /* space penalty if keep option */
                          /* is used */
}
```

This structure is defined in the *malloc.h* header file.

Each of the allocation routines returns a pointer to space suitably aligned (after possible pointer coercion) for storage of any type of object.

DIAGNOSTICS

Malloc, *realloc* and *calloc* return a NULL pointer if there is not enough available memory. When *realloc* returns NULL, the block pointed to by *ptr* is left intact. If *mallopt* is called after any allocation or if *cmd* or *value* are invalid, nonzero is returned. Otherwise, it returns zero.

RESTRICTIONS

This package usually uses more data space than *malloc*(3).
The code size is also bigger than *malloc*(3).
Note that unlike *malloc*(3), this package does not preserve the contents of a block when it is freed, unless the M_KEEP option of *mallopt* is used.
Undocumented features of *malloc*(3) have not been duplicated.

SEE ALSO

brk(2), *malloc*(3)

NAME

crl – VAX 8600 console RL02 interface

DESCRIPTION

This is a simple interface to the RL02 disk unit. which is part of the console subsystem for the VAX 8600. Access is given to the entire RL02 consisting of 512 cylinders of 2 tracks of 20 sectors of 256 bytes. The RL02 sectors are accessed as logical 512 byte disk blocks.

All I/O is raw; the seek addresses in raw transfers should be a multiple of 512 and a multiple of 512 bytes should be transferred, as in other 'raw' disk interfaces.

FILES

/dev/crl

SEE ALSO

arff(8V)

DIAGNOSTICS

crl: hard error sn%d crlcs=0x%b, crlds=0x%b

The console subsystem has reported a hard error while performing the requested I/O. 'clrcs' contains standard RLV12 control and status information and 'clrds' contains standard drive status information. Bit expansion in ascii is also provided.

crl: hndshk error

An error in communications between the console subsystem software and the Ultrix-32 operating system has occurred.

RESTRICTIONS

Only one "open" is allowed to the console RL02 device at any given time.

If a write is given with a count not a multiple of 512 bytes then the trailing portion of the last logical block will be zeroed.

The primary purpose of this driver is to apply updates to the console system disk. A 'block' interface is not provided.

NAME

cs – console diskette interface for VAX 8200 processor

DESCRIPTION

The *cs* special file is an interface to the RX50 diskette drive unit, which is accessed through a port on the 8200 processor board. Access is given to the entire diskette, consisting of 80 tracks with 10 sectors per track and 512 bytes per sector. Two units are supported and each can be accessed as a raw device (rcs1a or *rcs2a*) or as a block device (cs1a or *cs2a*). The *a* and *c* partitions are the only partitions provided and are equivalent.

FILES

/dev/cs1a /dev/cs2a	(block devices)
/dev/rcs1a /dev/rcs2a	(raw devices)

NAME

css – DEC IMP-11A LH/DH IMP interface

SYNTAX

pseudo-device imp
device css0 at uba0 csr 0167600 flags 10 vector cssrint cssxint

DESCRIPTION

The *css* device provides a Local Host/Distant Host interface to an IMP. It is normally used when participating in the DARPA Internet. The controller itself is not accessible to users, but instead provides the hardware support to the IMP interface described in *imp*(4). When configuring, the *imp* pseudo-device is also included.

DIAGNOSTICS

css%d: not alive. The initialization routine was entered even though the device did not autoconfigure. This is indicates a system problem.

css%d: can't initialize. Insufficient UNIBUS resources existed to initialize the device. This is likely to occur when the device is run on a buffered data path on an 11/750 and other network interfaces are also configured to use buffered data paths, or when it is configured to use buffered data paths on an 11/730 (which has none).

css%d: imp doesn't respond, icsr=%b. The driver attempted to initialize the device, but the IMP failed to respond after 500 tries. Check the cabling.

css%d: stray output interrupt csr=%b. An interrupt occurred when no output had previously been started.

css%d: output error, ocsr=%b icsr=%b. The device indicated a problem sending data on output.

css%d: recv error, csr=%b. The device indicated a problem receiving data on input.

css%d: bad length=%d. An input operation resulted in a data transfer of less than 0 or more than 1008 bytes of data into memory (according to the word count register). This should never happen as the maximum size of a host-IMP message is 1008 bytes.

reachable (the raw interface provides no routing support).

SEE ALSO

send(2), recv(2), intro(4N), pup(4F)

NAME

rl – RL02 moving head disk with RL11/RLV11 controller

SYNTAX

controller hl0 at uba? csr 0174400 vector rlintr
disk rl0 at hl0 drive 0

DESCRIPTION

Files with minor device numbers 0 through 7 refer to various portions of drive 0; minor devices 8 through 15 refer to drive 1, and so forth. The standard device names begin with *rl* followed by the drive number and then a letter a-h for partitions 0-7 respectively. The character *?* stands here for a drive number in the range 0-7.

The block files access the disk by the system's normal buffering mechanism and may be read and written without regard to physical disk records. There is also a raw interface, which provides for direct transmission between the disk and the user's read or write buffer. A single read or write call results in exactly one I/O operation and therefore raw I/O is considerably more efficient when many words are transmitted. The names of the raw files conventionally begin with an extra *r*.

Although RL02 disks have 256-byte sectors, the driver emulates 512-byte sectors. Raw I/O, counts should be a multiple of 512 bytes (a normal disk sector). Likewise, *seek* calls should specify a multiple of 512 bytes.

DISK SUPPORT

The origin and size (in 512-byte sectors) of the pseudodisks on each drive are as follows:

RL02 partitions:

disk	start	length	cyl
rl?a	0	15884	0-397
rl?b	15884	4520	398-510
rl?c	0	20480	0-511
rl?d	15884	4520	398-510
rl?g	0	20480	0-511

FILES

/dev/rl[0-7][a-h] (block files)
/dev/rrl[0-7][a-h] (raw files)

DIAGNOSTICS

The following messages are printed at the console:

rl%d%c: hard error sn%d.
An unrecoverable error occurred during transfer of the specified sector of the specified disk partition. Either the error was unrecoverable, or a large number of retry attempts (including offset positioning and drive recalibration) could not

recover the error. Additional register information may be gathered from the system error log file, */usr/adm/messages*.

rl%d: write protected.
The write protect switch was set on the drive when a write was attempted. The write operation is not recoverable.

hl%d: lost interrupt
A timer watching the controller detected no interrupt for an extended period while an operation was outstanding. This indicates a hardware or software failure. The error causes a UNIBUS reset and retry of the pending operations. If the controller continues to lose interrupts, this error will recur a few seconds later.

RESTRICTIONS

In raw I/O, *read*(2) and *write*(2) truncate file offsets to 512-byte block boundaries, and *write* scribbles on the tail of incomplete blocks. Thus, in programs that are likely to access raw devices, *read*(2), *write*(2) and *lseek*(2) should always deal in 512-byte multiples.

SEE ALSO

dmesg(8)

NAME

rx – DEC RX02 floppy disk interface

SYNTAX

controller fx0 at uba0 csr 0177170 vector rxintr
disk rx0 at fx0 drive 0
disk rx1 at fx0 slave 1

DESCRIPTION

The *rx* device provides access to a DEC RX02 floppy disk unit with M8256 interface module (RX211 configuration). The RX02 uses 8-inch, single-sided, soft-sectored floppy disks (with pre-formatted industry-standard headers) in either single or double density.

Floppy disks handled by the RX02 contain 77 tracks, each with 26 sectors (for a total of 2,002 sectors). The sector size is 128 bytes for single density, 256 bytes for double density. Single density disks are compatible with the RX01 floppy disk unit and with IBM 3740 Series Diskette 1 systems.

In addition to normal ('block' and 'raw') i/o, the driver supports formatting of disks for either density and the ability to invoke a 2 for 1 interleaved sector mapping compatible with the DEC operating system RT-11.

The minor device number is interpreted as follows:

Bit	**Description**
0	Sector interleaving (1 disables interleaving)
1	Logical sector 1 is on track 1 (0 no, 1 yes)
2	Not used, reserved
Other	Drive number

The two drives in a single RX02 unit are treated as two disks attached to a single controller. Thus, if there are two RX02's on a system, the drives on the first RX02 are "rx0" and "rx1", while the drives on the second are "rx2" and "rx3".

When the device is opened, the density of the disk currently in the drive is automatically determined. If there is no floppy in the device, open will fail.

The interleaving parameters are represented in raw device names by the letters 'a' through 'd'. Thus, unit 0, drive 0 is called by one of the following names:

Mapping	**Device name**	**Starting track**
interleaved	/dev/rrx0a	0
direct	/dev/rrx0b	0
interleaved	/dev/rrx0c	1
direct	/dev/rrx0d	1

The mapping used on the 'c' device is compatible with the DEC operating system RT-11. The 'b' device accesses the sectors of the disk in strictly sequential order. The 'a' device is the most efficient for disk-to-disk copying.

NAME

tar, mdtar – tape archive file format

DESCRIPTION

The tape archive command *tar*(5) dumps several files, including special files, into one, in a medium suitable for transportation.

A *tar* tape or file is a series of blocks. Each block is of size TBLOCK. A file on the tape is represented by a header block, which describes the file, followed by zero or more blocks, which give the contents of the file. At the end of the tape are two blocks filled with binary zeros, as an end-of-file indicator.

The blocks are grouped for physical I/O operations. Each group of *n* blocks (where *n* is set by the **b** keyletter on the *tar*(1) command line, and the default is 20 blocks) is written with a single system call; on 9-track tapes, the result of this write is a single tape record. The last group is always written at the full size, so blocks after the two zero blocks contain random data. On reading, the specified or default group size is used for the first read, but if that read returns less than a full tape block, the reduced block size is used for further reads.

The header block looks like:

```
#define TBLOCK  512
#define NAMSIZ  100

union hblock {
        char dummy[TBLOCK];
        struct header {
                char name[NAMSIZ];
                char mode[8];
                char uid[8];
                char gid[8];
                char size[12];
                char mtime[12];
                char chksum[8];
                char linkflag;
                char linkname[NAMSIZ];
                char rdev[6]
        } dbuf;
};
```

The *name* field is a null-terminated string. The other fields are zero-filled octal numbers in ASCII. Each field (of width *w*) contains *w* minus 2 digits, a space, and a null, except *size* and *mtime*, which do not contain the trailing null. The *name* field specifies the name of the file, as specified on the *tar* command line. Files dumped because they were in a directory that was named in the command line have the directory name as prefix and */filename* as suffix. The *mode* field specifies the file mode, with the top bit masked off. The *uid* and *gid* fields specify the user and group numbers that own the file. The *size* field specifies the size of the file in bytes. Links and symbolic links are dumped with this field specified as zero. The *mtime* field specifies the modification time of the file at the time it was dumped. The *chksum* field is a decimal ASCII value, which represents the sum of all the bytes in the header block. When calculating the checksum, the *chksum* field is treated as if it were all blanks. The *linkflag* field is ASCII 0 if the file is normal or a special file and ASCII 1 if it is a hard link, and ASCII 2 if it is a symbolic link. The name to which it is linked, if any, is in *linkname*, with a trailing null. Unused fields of the header are binary zeros (and are included in the checksum). The *rdev* field encodes the ASCII representation of a device special file's major and minor device numbers.

The first time a given i-node number is dumped, it is dumped as a regular file. The second and subsequent times, it is dumped as a link instead. Upon retrieval, if a link entry is retrieved, but not the file it was linked to, an error message is printed and the tape must be manually rescanned to retrieve the linked file.

The encoding of the header is designed to be portable across machines.

RESTRICTIONS

Names or link names longer than NAMSIZ produce error reports and cannot be dumped.

SEE ALSO

tar(1)

HOW TO ORDER ADDITIONAL DOCUMENTATION

DIRECT TELEPHONE ORDERS

In Continental USA and Puerto Rico call **800–258–1710**	In Canada call **800–267–6146**	In New Hampshire, Alaska or Hawaii call **603–884–6660**

DIRECT MAIL ORDERS (U.S. and Puerto Rico*)

DIGITAL EQUIPMENT CORPORATION
P.O. Box CS2008
Nashua, New Hampshire 03061

DIRECT MAIL ORDERS (Canada)

DIGITAL EQUIPMENT OF CANADA LTD.
940 Belfast Road
Ottawa, Ontario, Canada K1G 4C2
Attn: A&SG Business Manager

INTERNATIONAL

DIGITAL EQUIPMENT CORPORATION
A&SG Business Manager
c/o Digital's local subsidiary
or approved distributor

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Westminster, Massachusetts 01532

*Any prepaid order from Puerto Rico must be placed with the Local Digital Subsidiary:
809–754–7575

ULTRIX-32
Version 1.2
Release Notes
AA-BG57C-TE

Reader's Comments

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement. ______________________________

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) ______________________________

Name ______________________________ Date ______________________________

Organization ______________________________

Street ______________________________

City ______________________________ State ______ Zip Code or Country ______________

------Do Not Tear - Fold Here and Tape --

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO 33 MAYNARD MASS

POSTAGE WILL BE PAID BY ADDRESSEE

Documentation Manager
ULTRIX Documentation Group
MK02-1/H10
Continental Blvd.
Merrimack, N.H. 03054

------Do Not Tear - Fold Here and Tape --

Cut Along Dotted Line