305-512
Issue 1

AT&T 3B2 Computer
UNIX™ System V Release 2.0

System Administration Reference Manual

NOTICE

The information in this document is subject to change without notice. AT&T assumes no responsibility for any errors that may appear in this document.

TRADEMARKS

The following is a listing of the trademarks that are used in this manual:

- UNIX — Trademark of AT&T
- DOCUMENTER'S WORKBENCH — Trademark of AT&T
- DIABLO — Registered trademark of Xerox Corporation
- HP — Trademark of Hewlett-Packard, Inc.
- Versatec — Trademark of Versatec Corporation
- TELETYPE — Registered trademark of AT&T
- DEC, PDP, VAX, UNIBUS, and MASSBUS — Trademarks of Digital Equipment Corporation
- TEKTRONIX — Registered trademark of Tektronic, Inc.
- WREN — Trademark of Control Data Corporation

ORDERING INFORMATION

Additional copies of this document can be ordered by calling

1-800-432-6600 Inside the U.S.A.

OR

1-317-352-8557 Outside the U.S.A.

OR by writing to:

AT&T Customer Information Center (CIC)
Attn: Customer Service Representative
P.O. Box 19901
Indianapolis, IN 46219

Replace this

page with the

Introduction

tab separator.

INTRODUCTION

This manual is intended to supplement the information contained in the *AT&T 3B2 Computer User Reference Manual* and to provide an easy reference volume for those who must administer a UNIX system.

This manual is divided into three sections:

1. System Maintenance Commands and Application Programs
7. Special Files
8. System Maintenance Procedures

Throughout this volume, each reference of the form *name*(1M), *name*(7), or *name*(8), refers to entries in this manual, while all other references to entries of the form *name*(N), where *N* is a number possibly followed by a letter, refer to entry *name* in Section *N* of the *AT&T 3B2 Computer User Reference Manual* or the *AT&T 3B2 Computer Programmer Reference Manual.*

Section 1 (*System Maintenance Commands and Application Programs*) contains a subset of commands that are initially resident on the hard disk (labeled Essential Utilities) and commands that are installable from the System Administration Utilities floppy diskette. These entries carry a sub-class designation of "1M" for cross-referencing reasons.

Section 7 (*Special Files*) discusses the characteristics of each system file that actually refers to an input/output device. The names in this section generally refer to device names for the hardware, rather than to the names of the special files themselves.

Section 8 (*System Maintenance Procedures*) discusses crash recovery and boot procedures, facility descriptions, etc.

Each section consists of a number of independent entries of a page or so each. The name of the entry appears in the upper corners of its pages. Entries within each section are alphabetized, with the exception of the introductory entry that begins each section. Some entries may describe several routines, commands, etc. In such cases, the entry appears only once, alphabetized under its "major" name.

All entries are based on a common format, not all of whose parts always appear:

The NAME part gives the name(s) of the entry and briefly states its purpose.

The SYNOPSIS part summarizes the use of the program being described. A few conventions are used, particularly in Section 1 (*Commands*):

Boldface strings are literals and are to be typed just as they appear.

Italic strings usually represent substitutable argument prototypes and program names found elsewhere in the manual. (They are underlined in the typed version of the entries.)

Square brackets [] around an argument prototype indicate that the argument is optional. When an argument prototype is given as "name" or "file", it always refers to a *file* name.

Ellipses ... are used to show that the previous argument prototype may be repeated.

A final convention is used by the commands themselves. An argument beginning with a minus −, plus +, or equal sign = is often taken to be some sort of flag argument, even if it appears in a position where a file name could appear. Therefore, it is unwise to have files whose names begin with −, +, or =.

The DESCRIPTION part discusses the subject at hand.

The **EXAMPLE(S)** part gives example(s) of usage, where appropriate.

The **FILES** part gives the file names that are built into the program.

The **SEE ALSO** part gives pointers to related information.

The **DIAGNOSTICS** part discusses the diagnostic indications that may be produced. Messages that are intended to be self-explanatory are not listed.

The **WARNINGS** part points out potential pitfalls.

The **BUGS** part gives known bugs and sometimes deficiencies. Occasionally, the suggested fix is also described.

A table of contents and a permutted index derived from that table precede section 1M. On each *index* line, the title of the entry to which that line refers is followed by the appropriate section number in parentheses. The *Permuted Index* is used by searching the middle column for a key word or phrase. The right column will then contain the name of the manual page that contains that command. The left column contains additional useful information about the command.

Replace this

page with the

Table of Contents

tab separator.

TABLE OF CONTENTS

1. System Maintenance Commands and Application Programs

7. Special Files

8. System Maintenance Procedures

Replace this

page with the

Permuted Index

tab separator.

Replace this

page with the

Section 1 (Maint. Commands)

tab separator.

NAME

intro – introduction to system maintenance commands and application programs

DESCRIPTION

This section describes, in alphabetical order, commands that are used chiefly for system maintenance and administration purposes. The commands in this section should be used along with those listed in Section 1 of the *AT&T 3B2 Computer User Reference Manual* and Sections 2, 3, 4, and 5 of the *AT&T 3B2 Computer Programmer Reference Manual*. References to other manual entries not of the form *name*(1M), *name*(7) or *name*(8) refer to entries of the above manuals.

COMMAND SYNTAX

Unless otherwise noted, commands described in this section accept options and other arguments according to the following syntax:

name [*option*(*s*)] [*cmdarg*(*s*)]
where:

name	The name of an executable file.
option	– *noargletter*(*s*) or, – *argletter*<>*optarg* where <> is optional white space.
noargletter	A single letter representing an option without an argument.
argletter	A single letter representing an option requiring an argument.
optarg	Argument (character string) satisfying preceding *argletter*.
cmdarg	Path name (or other command argument) *not* beginning with – or, – by itself indicating the standard input.

SEE ALSO

getopt(1) in the *AT&T 3B2 Computer User Reference Manual*.
getopt(3C) in the *AT&T 3B2 Computer Programmer Reference Manual*.

DIAGNOSTICS

Upon termination, each command returns two bytes of status, one supplied by the system and giving the cause for termination, and (in the case of "normal" termination) one supplied by the program (see *wait*(2) and *exit*(2)). The former byte is 0 for normal termination; the latter is customarily 0 for successful execution and nonzero to indicate troubles such as erroneous parameters, bad or inaccessible data, or other inability to cope with the task at hand. It is called variously "exit code", "exit status", or "return code", and is described only where special conventions are involved.

BUGS

Regretfully, many commands do not adhere to the aforementioned syntax.

NAME

accept, reject — allow/prevent LP requests

SYNOPSIS

/usr/lib/accept destinations
/usr/lib/reject [−r[reason]] destinations

DESCRIPTION

Accept allows *lp*(1) to accept requests for the named *destinations*. A *destination* can be either a printer or a class of printers. Use *lpstat*(1) to find the status of *destinations*.

Reject prevents *lp*(1) from accepting requests for the named *destinations*. A *destination* can be either a printer or a class of printers. Use *lpstat*(1) to find the status of *destinations*. The following option is useful with *reject*.

−r[*reason*] Associates a *reason* with preventing *lp* from accepting requests. This *reason* applies to all printers mentioned up to the next −r option. *Reason* is reported by *lp* when users direct requests to the named *destinations* and by *lpstat*(1). If the −r option is not present or the −r option is given without a *reason*, then a default *reason* will be used.

FILES

/usr/spool/lp/*

SEE ALSO

lpadmin(1M), lpsched(1M).
enable(1), lp(1), lpstat(1) in the *AT&T 3B2 Computer User Reference Manual.*

NAME

brc, bcheckrc, rc2 – system initialization shell scripts

SYNOPSIS

/etc/brc

/etc/bcheckrc

/etc/rc2

DESCRIPTION

These shell procedures are executed via entries in **/etc/inittab** by *init*(1M) when the system changes run states.

The *brc* procedure clears the mounted file system table, **/etc/mnttab** and puts root into the mount table.

The *bcheckrc* procedure checks the status of the root file system. If the root file system is found to be bad, *bcheckrc* repairs it.

The *rc2* procedure executes **/etc/TIMEZONE** and every file in the **rc.d** directory.

These shell procedures, may be used for several run-level states. The *who*(1) command may be used with the –r argument to get the run-level informantion.

FILES

/etc/rc.d

SEE ALSO

fsck(1M), init(1M), shutdown(1M).
who(1) in the *AT&T 3B2 Computer User Reference Manual.*

NAME

checkall — faster file system checking procedure

SYNOPSIS

/etc/checkall

DESCRIPTION

The *checkall* procedure is a prototype and must be modified to suit local conditions. The following will serve as an example:

```
# check the root file system by itself
fsck /dev/dsk/c1d0s6

# dual fsck of (integral hard disk)
dfsck /dev/rdsk/c1d0s2 — /dev/rdsk/c1d1s1
```

The *checkall* procedure takes 11 minutes.

Dfsck is a program that permits an operator to interact with two *fsck*(1M) programs at once. To aid in this, *dfsck* will print the file system name for each message to the operator. When answering a question from *dfsck*, the operator must prefix the response with a **1** or a **2** (indicating that the answer refers to the first or second file system group).

Due to the file system load balancing required for dual checking, the *dfsck* command should always be executed through the *checkall* shell procedure.

In a practical sense, the file systems are divided as follows:

```
dfsck file_systems_on_drive_0 — file_systems_on_drive_1
  . . .
```

WARNINGS

1. Do not use *dfsck* to check the *root* file system.
2. The *dfsck* procedure is useful only if the system is set up for multiple physical I/O buffers.

SEE ALSO

fsck(1M).

NAME

checkfsys – check a file system on a removable disk

SYNOPSIS

The *checkfsys* command allows the user to check for and optionally repair a damaged file system on a removable disk.

The user is asked if they wish to:

check the file system
: No repairs are attempted.

repair it interactively
: The user is informed about each instance of damage and asked if they wish to repair it.

repair it automatically
: The program applies a standard repair to each instance of damage.

This command may be controlled by passwords. See *sysadm*(1), the **admpasswd** sub-command.

The identical function is also available under the *sysadm* menu:

sysadm checkfsys

WARNING

While automatic and interactive checks are generally successful, they can occasionally lose a file or a file's name. Files with content but without names are put in the */file-system/***lost+found** directory.

If losing data is of particular concern, "check" the file system first to discover how damaged it appears to be. Then either use one of the repair mechanisms or get knowledgeable help from someone who knows how to debug and reconstruct a file system.

SEE ALSO

fsck(1M), makefsys(1M), mountfsys(1M).
sysadm(1) in the *AT&T 3B2 Computer User Reference Manual.*

NAME

chroot – change root directory for a command

SYNOPSIS

/etc/chroot newroot command

DESCRIPTION

The given command is executed *relative to the new root*. The meaning of any initial slashes (/) in path names is changed for a command and any of its children to *newroot*. Furthermore, the initial working directory is *newroot*.

Notice that:

chroot newroot command >x

will create the file x relative to the original root, not the new one.

This command is restricted to the super-user.

The new root path name is always relative to the current root: even if a *chroot* is currently in effect, the *newroot* argument is relative to the current root of the running process.

SEE ALSO

cd(1) in the *AT&T 3B2 Computer User Reference Manual.*

BUGS

One should exercise extreme caution when referencing special files in the new root file system.

NAME

ckauto – determine if the UNIX operating system was reconfigured at boot time.

SYNOPSIS

/etc/ckauto

DESCRIPTION

The *ckauto* command uses a *sys3b*(2) call to determine if the UNIX system was reconfigured during initialization. If the *sys3b* call is successful, *ckauto* exits with 1, otherwise it exists with 0. *Ckauto* is intended to be used by utilities that will do services based on the output of the *sys3b* call.

SEE ALSO

sys3b(2) in the *AT&T 3B2 Computer Programmer Reference Manual.*

NAME

/etc/ckbupscd – check backup schedule

SYNOPSIS

/etc/ckbupscd [**–m**]

DESCRIPTION

Ckbupscd consults the file **/etc/bupsched** and prints the file system lists from lines with date and time specifications matching the current time. If the **–m** flag is present an introductory message in the output is suppressed so that only the file system lists are printed.

The Simple administration commands *bupsched/schedcheck* are provided to review and edit the **/etc/bupsched** file.

The file **/etc/bupsched** should contain lines of 4 or more fields, separated by spaces or tabs. The first 3 fields (the schedule fields) specify a range of dates and times. The rest of the fields constitute a list of names of file systems to be printed if *ckbupscd* is run at some time within the range given by the schedule fields. The general format is:

time[,time] day[,day] month[,month] fsyslist

where:

time Specifies an hour of the day (*0* through *23*), matching any time within that hour, or an exact time of day (*0:00* through *23:59*).

day Specifies a day of the week (*sun* through *sat*) or day of the month (*1* through *31*).

month Specifies the month in which the time and day fields are valid. Legal values are the month numbers (*1* through *12*).

fsyslist The rest of the line is taken to be a file system list to print.

Multiple time, day, and month specifications may be separated by commas, in which case they are evaluated left to right.

An asterisk (*) always matches the current value for that field.

A line beginning with a sharp sign (#) is interpretted as a comment and ignored.

The longest line allowed (including continuations) is 1024 characters.

EXAMPLES

The following are examples of lines which could appear in the **/etc/bupsched** file.

06:00-09:00 fri 1,2,3,4,5,6,7,8,9,10,11 /applic
Prints the file system name */applic* if *ckbupscd* is run between 6:00am and 9:00am any Friday during any month except December.

00:00-06:00,16:00-23:59 1,2,3,4,5,6,7 1,8 /
Prints a reminder to backup the root (*/*) file system if *ckbupscd* is run between the times of 4:00pm and 6:00am during the first week of August or January.

SEE ALSO

cron(1M).
echo(1), sh(1), sysadm(1) in the *AT&T 3B2 Computer User Reference Manual.*

FILES

/etc/bupsched specification file containing times and file system to back up

BUGS

Ckbupscd will report file systems due for backup if invoked any time in the window. It does not know that backups may have just been taken.

NAME

clri – clear i-node

SYNOPSIS

/etc/clri file-system i-number ...

DESCRIPTION

Clri writes zeros on the 64 bytes occupied by the i-node numbered *i-number*. *File-system* must be a special file name referring to a device containing a file system. After *clri* is executed, any blocks in the affected file will show up as "missing" in an *fsck*(1M) of the *file-system*.

Read and write permission is required on the specified *file-system* device. The i-node becomes allocatable.

The primary purpose of this routine is to remove a file which for some reason appears in no directory. If it is used to *zap* an i-node which does appear in a directory, care should be taken to track down the entry and remove it. Otherwise, when the i-node is reallocated to some new file, the old entry will still point to that file. At that point removing the old entry will destroy the new file. The new entry will again point to an unallocated i-node, so the whole cycle is likely to be repeated again and again.

SEE ALSO

fsck(1M), fsdb(1M), ncheck(1M).
fs(4) in the *AT&T 3B2 Computer Programmer Reference Manual.*

BUGS

If the file is open, *clri* is likely to be ineffective.

WARNINGS

This command should only be used in emergencies and extreme care should be exercised.

NAME

/etc/cmpress — re-link file system to remove fragmentation

SYNOPSIS

/etc/cmpress

DESCRIPTION

Cmpress re-links the input file system to improve access time by cleaning up fragmentation of files throughout the file system. The file system must be mounted in order for this procedure to find the file system and determine its characteristics.

Cmpress uses a 3B2 Computer cartridge tape for intermediate storage. The file system is first copied onto the tape, the old file system is removed and the free block list is sorted into sequential order, then the file system is copied back onto the disk so that file system blocks that previously were scattered throughout the file system are in contiguous space.

Notice that the file system is destroyed during the process of compressing it. For this reason it is strongly recommended that an up-to-date backup of the file system be made before the file system is compressed. In the event of a mishap during file system compression the file system could be restored from the backup.

Since the file system is destroyed during the compression process, it is not possible to compress the root file system. The *cmpress* command will reject the file system name / if it is entered.

Compressing any file system except **/usr** can be done through the *sysadm* command. An example of such a file compression is given below. Compressing the **/usr** file system is somewhat more complex a process, since the *sysadm* facilities reside in the **/usr** file system. A scheme for compressing **/usr** is given in the examples.

EXAMPLES

To compress a file system named *applic*, the following command would be issued:

sysadm tapemgmt

When the tape management facilities menu is displayed, select the *compress* facility and answer the questions posed by the shell procedure. This *compress* facility invokes the **/etc/cmpress** procedure.

To compress the **/usr** file system the UNIX system has to be in single user mode with the **/usr** file system mounted. The following sequence of commands will take the system from multiuser to single user mode, compress the **/usr** file system, then return the system to multiuser mode. Notice that any work going on in the system at the time that the system is changed to single user mode will be terminated, so this process should be done at a time when there are no other users logged in, and no background tasks are being done. It should be done only from the console from the root login.

First check to see how **/usr** is mounted.

mount

Make note of the **/dev/rSA/c?d?s?** information, as you will need it later.

Now take the system down to single user mode.

init 1

Lots of messages will now appear, and you will need to log back in as root.

Now mount the **/usr** file system. Use the **/dev/rSA/c?d?s?** information from the *mount* command above.

```
mount /dev/dsk/c1d0s2 /usr
```

Now compress the file system

```
/etc/cmpress
```

The procedure will pose a series of questions. As the compression process runs it will display a series of messages indicating its progress.

When compression is complete the following commands will unmount the **/usr** file system and return the system to multi user mode. Many messages will be displayed during this process.

```
umount /dev/dsk/c1d0s2
init 2
```

SEE ALSO

sysadm(1) in the *AT&T 3B2 Computer User Reference Manual.*

DIAGNOSTICS

The diagnostic messages are intended to be self explanatory.

WARNINGS

As mentioned above, since the compression of the file system entails its destruction and restoration it is strongly recommended that a backup copy of the file system be made before its compression is attempted.

NAME

crash – examine system images

SYNOPSIS

/etc/crash [system_image] [namelist]

DESCRIPTION

The *crash* command is used to interactively examine system memory images. In response to user requests, *crash* formats and prints system control structures and other miscellaneous information.

Parameters to crash are the *system_image*, the file containing the system memory image and *namelist*, the text file that was booted to produce the *system*memory*image*. This file contains the symbol table information needed for symbolic access to the system memory image.

System_image is defaulted to **/dev/mem** or the active memory of a running system. This can also be specified by '–'. The system image can also be **/dev/ifdsk06**, with the first floppy of a system dump taken with *sysdump*(8) or the path name of a file produced by *ldsysdump*(1M).

The default *namelist* is the **/unix** of the current machine. If a system image from another machine is to be used, the text file must be copied from that machine as well.

Commands

Input to *crash* is typically of the form:

command [options] [items]

where *options* modifies the format of the output and *items* indicates which items are printed. If a table is being dumped, the default is all valid table entries. List items are specified by table entry number (0 based). Default for items that are process related is the current process, i.e., the process that was running at the time of the crash.

The use of [a] denotes that "a" is optional, (a\|\b) that either "a" or "b" are acceptable, but not both.

The commands are:

? print synopsis of commands

!*cmd* escape to shell to execute a command.

buffer [format] [list of buffers]
Aliases: **b**
Print the data from a system buffer according to *format*. If no format is specified, the last format is used. Valid formats are **byte**, **character**, **decimal**, **directory**, **hex**, **inode**, **octal**, and **write**. The **write** format creates a file **buf.#** in the current directory containing the buffer data.

bufhdr [list of buffers]
Aliases: **buf**, **hdr**
Format and print system buffer headers.

callout Aliases: **c**, **call**, **calls**, **time**, **timeout**, **tout**
Print all entries in the callout table.

ds data_address
Print data or bss symbol closest to but less than *data_address*.

file [list of file table entries]
Aliases: **i**, **ino**
Format and print i-node table entries.

kfp [address]
Aliases: **fp**, **r9**
Prints stack frame pointer of process that caused panic. If *address* is specified, sets frame pointer to address.

lck
Aliases: **l**
Print the active and sleep record lock tables; also verify the correctness of the record locking linked lists.

map
Formats and prints coremap.

mount [list of mount table entries]
Aliases: **m**, **mnt**
Format and print mount table entries.

nm symbol
Print address and type of specified symbol.

nvram area
Format and print contents of nvram areas. Areas are: **fwnvr**, **unxnvr**, **systate**, and **errlog**.

od (symbol | [−p] address) [count [format]]
Aliases: **dump**, **rd**
Dump *count* data values starting at symbol value or address according to given *format*. A physical address is indicated by '−p'.

pcb [list of process table entries]

pcb −i [symbol| address [index]]
Format and print process control blocks. The **−i** indicates an interrupt pcb. *Index* is used when an array of pcbs is addressed.

proc [− [r]] [list of process table entries]
Aliases: **p**, **ps**
Format and print process table entries. The '−' option generates more information. The '−r' option limits output to runnable processes.

quit
Aliases: **q**
Terminates the crash session.

regs
Prints the MMU registers.

sdt [− segment table entry] [list of process table entries]
Prints the segment descriptor tables for sections 2 and 3. The *−segment table entry* limits printing to the specified table entry.

smap
Formats and prints swap map.

sram
Prints SRAM values.

stack [list of process table entries]

stack −i (symbol| address) [index]
Aliases: **k**, **kernel**, **s**, **stk**
Dump the process stack. The **−i** option specifed interrupt stacks. *Index* is used with arrays of interrupt pcbs.

stat Print system statistics.

sysdt [k | K | m | M]
Formats and prints kernel segment descriptor tables (sections 0 and 1). **K** or **k** restricts printing to section 0. **M** or **m** restricts printing to section 1.

tables [list of process table entries]
Aliases: **tbls**
Prints segment descripto tables for section 2 and 3.

text [list of text table entries]
Aliases: **txt, x**
Format and print text table entries.

trace [−r] [process table entries]

trace [−r] −**i** (**symbol** | **address**) [**index**]
Aliases: **t**
Trace provides a formatted dump of the specified stack. The −**i** **options specifies an interrupt stack.** Index is used with arrays of interrupt pcbs.

ts text-address
Prints the text symbol closest to but less than the text address.

tty [type] [−i] [list of tty table entries]
Aliases: **con, term**
Formats and prints stty tables. The −**i** option causes stty settings to be printed.

user [list of process table entries]
Aliases: **u, u_area, uarea, ublock**
Formats and prints ublocks.

var Aliases: **tunable, tunables, tune, v**
Prints tunable system parameters.

FILES

buf.#	file in current dirctory where beffer number # is written
/dev/mem	system image of currently running system
/dev/ifdsk06	used to access system image on floppy diskette

SEE ALSO
ldsysdump(1M), sysdump(8).

NAME

cron - clock daemon

SYNOPSIS

/etc/cron

DESCRIPTION

Cron executes commands at specified dates and times. Regularly scheduled commands can be specified according to instructions found in crontab files; users can submit their own crontab file via the *crontab* command. Commands which are to be executed only once may be submitted via the *at* command. Since *cron* never exits, it should only be executed once. This is best done by running *cron* from the initialization process through the file **/etc/rc2**.

Cron only examines crontab files and at command files during process initialization and when a file changes. This reduces the overhead of checking for new or changed files at regularly scheduled intervals.

FILES

/usr/lib/cron	main cron directory
/usr/lib/cron/log	accounting information
/usr/spool/cron	spool area

SEE ALSO

at(1), crontab(1), sh(1) in the *AT&T 3B2 Computer User Reference Manual.*

DIAGNOSTICS

A history of all actions taken by cron are recorded in **/usr/lib/cron/log.**

NAME

ctccpio – copy file archives in and out to cartridge tape

SYNOPSIS

ctccpio **–o** [**a|v|V|K**] **–T** cartridge_tape_device

ctccpio **–i** [**dmrtu|v|V|f**] **–T** cartridge_tape_device [patterns]

DESCRIPTION

Ctccpio **–o** (copy out) reads the standard input to obtain a list of path names and copies those files together with path name and status information to the cartridge tape device specified on the command line.

Ctccpio **–i** (copy in) extracts files from the cartridge tape device specified on the command line, which is assumed to be the product of a previous **ctccpio** **–o**. Only files with names that match *patterns* are selected. *Patterns* are given in the name-generating notation of *sh*(1). In *patterns*, meta-characters ?, *, and [...] match the slash / character. Multiple *patterns* may be specified and if no *patterns* are specified, the default for *patterns* is * (i.e., select all files). The extracted files are conditionally created and copied into the current directory tree based upon the options described below. The permissions of the files will be those of the previous **ctccpio** **–o**. The owner and group of the files will be that of the current user unless the user is super-user, which causes *ctccpio* to retain the owner and group of the files of the previous **ctccpio** **–o**.

Ctccpio is a modified version of cpio that knows how to use the streaming feature of the cartridge tape controller. *Ctccpio* will provide better performance than cpio when using the cartridge tape, although the amount of streaming that is achieved will depend upon the size of the files to be copied, file system fragmentation, etc. Headers are always written in ASCII character format.

The meanings of the available options are:

a Reset access times of input files after they have been copied.

d *Directories* are to be created as needed.

r Interactively *rename* files. If the user types a null line, the file is skipped.

t Print a *table of contents* of the input. No files are created.

u Copy *unconditionally* (normally, an older file will not replace a newer file with the same name).

v *Verbose*: causes a list of file names to be printed. When used with the **t** option, the table of contents looks like the output of an **ls** **–l** command [see *ls*(1)]. (Do not use with V option.)

V Print out a dot for each file copied. (Do not use with **v** option.)

m Retain previous file modification time. This option is ineffective on directories that are being copied.

f Copy in all files except those in *patterns*.

K Perform verify pass on output to tape. This option will significantly decrease the performance of *ctccpio*.

EXAMPLES

The first example below copies the contents of a directory to the tape device **/dev/rSA/ctape1**; the second reads the tape and writes the files copied in the first example:

```
cd olddir
find . -print | ctccpio -oT /dev/rSA/ctape1

cd newdir
ctccpio -iT /dev/rSA/ctape1
```

SEE ALSO

ar(1), find(1), ls(1) in the *AT&T 3B2 Computer User Reference Manual.*
cpio(4) in the *AT&T 3B2 Computer Programmer Reference Manual.*

WARNING

Only partition 3 is recommended for streaming. Use of other partitions that begin at the tape's logical block 0 will corrupt tape format data and require re-formatting the tape.

BUGS

Path names are restricted to 128 characters. If there are too many unique linked files, the program runs out of memory to keep track of them and, thereafter, linking information is lost. Only the super-user can copy special files.

NAME

/etc/ctcfmt – format cartridge tape

SYNOPSIS

/etc/ctcmft [**–v**] [**–p** passct] **–t** rawdevice

DESCRIPTION

Ctcfmt is used to format or reformat a tape cartridge on the 3B2 Computer. The parameter *rawdevice* specifies the name of the device, as a raw device name, of the drive to be used for the formatting operation.

The option **–p** indicates that the maximum number of tape passes to be allowed for the tape is to be set to *passct*. This number is used to warn the user that the tape is approaching the end of its lifetime and should be replaced to avoid the loss of data. The default value of *passct* is 4000.

The option **–v** indicates that the format is to be verified after being done. Use of this option will prevent a detectably bad tape from being used, since verification will read the format information from every block on the tape and detect unreadable blocks. Because it makes two passes over the medium and checks every block on it, the time needed to format a medium with this option is noticably longer than without this option. The decision of whether or not to verify should be based on the importance of the data to be placed on the medium, and on the degree to which the medium currently being used by the user exhibits defects.

EXAMPLES

The following command formats a floppy tape with 4000 as the maximum pass count and does not verify the formatting operation:

```
ctcfmt –t /dev/rSA/ctape1
```

The following command does a format with verification and sets the pass count to 3500:

```
ctcfmt –v –p 3500 –t /dev/rSA/ctape2
```

DIAGNOSTICS

The diagnostic messages are intended to be self explanatory.

WARNINGS

Once a tape has been formatted, any data previously on the medium is lost.

NAME

dcopy – copy file systems for optimal access time

SYNOPSIS

/etc/dcopy [–sX] [–an] [–d] [–v] [–ffsize[:isize]] inputfs outputfs

DESCRIPTION

Dcopy copies file system *inputfs* to *outputfs*. *Inputfs* is the existing file system; *outputfs* is an appropriately sized file system, to hold the reorganized result. For best results *inputfs* should be the raw device and *outputfs* should be the block device. *Dcopy* should be run on unmounted file systems (in the case of the root file system, copy to a new pack). With no arguments, *dcopy* copies files from *inputfs* compressing directories by removing vacant entries, and spacing consecutive blocks in a file by the optimal rotational gap. The possible options are

–s*X* supply device information for creating an optimal organization of blocks in a file. The forms of *X* are the same as the **–s** option of *fsck*(1M).

–a*n* place the files not accessed in *n* days after the free blocks of the destination file system (default for *n* is 7). If no *n* is specified then no movement occurs.

–d leave order of directory entries as is (default is to move sub-directories to the beginning of directories).

–v currently reports how many files were processed, and how big the source and destination freelists are.

–f*fsize*[*:isize*]
specify the *outputfs* file system and inode list sizes (in blocks). If the option (or *:isize*) is not given, the values from the *inputfs* are used.

Dcopy catches interrupts and quits and reports on its progress. To terminate *dcopy* send a quit signal, and *dcopy* will no longer catch interrupts or quits.

SEE ALSO

fsck(1M), mkfs(1M).
ps(1) in the *3B2 Computer System User Reference Manual.*

NAME

dd – convert and copy a file

SYNOPSIS

dd [option=value] ...

DESCRIPTION

Dd copies the specified input file to the specified output with possible conversions. The standard input and output are used by default. The input and output block size may be specified to take advantage of raw physical I/O.

option	*values*
if=*file*	input file name; standard input is default
of=*file*	output file name; standard output is default
ibs=*n*	input block size *n* bytes (default 512)
obs=*n*	output block size (default 512)
bs=*n*	set both input and output block size, superseding *ibs* and *obs*; also, if no conversion is specified, it is particularly efficient since no in-core copy need be done
cbs=*n*	conversion buffer size
skip=*n*	skip *n* input blocks before starting copy
seek=*n*	seek *n* blocks from beginning of output file before copying
count=*n*	copy only *n* input blocks
conv=**ascii**	convert EBCDIC to ASCII
ebcdic	convert ASCII to EBCDIC
ibm	slightly different map of ASCII to EBCDIC
lcase	map alphabetics to lower case
ucase	map alphabetics to upper case
swab	swap every pair of bytes
noerror	do not stop processing on an error
sync	pad every input block to *ibs*
..., ...	several comma-separated conversions

Where sizes are specified, a number of bytes is expected. A number may end with **k**, **b**, or **w** to specify multiplication by 1024, 512, or 2, respectively; a pair of numbers may be separated by **x** to indicate a product.

Cbs is used only if *ascii* or *ebcdic* conversion is specified. In the former case *cbs* characters are placed into the conversion buffer, converted to ASCII, and trailing blanks trimmed and new-line added before sending the line to the output. In the latter case ASCII characters are read into the conversion buffer, converted to EBCDIC, and blanks added to make up an output block of size *cbs*.

After completion, *dd* reports the number of whole and partial input and output blocks.

SEE ALSO

cp(1) in the *AT&T 3B2 Computer User Reference Manual.*

DIAGNOSTICS

f+p blocks in(out) numbers of full and partial blocks read(written)

NAME

devnm – device name

SYNOPSIS

/etc/devnm [names]

DESCRIPTION

Devnm identifies the special file associated with the mounted file system where the argument *name* resides. (As a special case, both the block device name and the swap device name are printed for the argument name / if swapping is done on the same disk section as the **root** file system.) Argument names must be full path names.

This command is most commonly used by **/etc/brc** (see *brc*(1M)) to construct a mount table entry for the **root** device.

EXAMPLE

The command:

```
/etc/devnm /usr
```

produces

```
/dev/c1d0s2 usr
```

if **/usr** is mounted on **/dev/dsk/c1d0s2**.

FILES

/dev/dsk/*
/etc/mnttab

SEE ALSO

brc(1M).

NAME

df – report number of free disk blocks

SYNOPSIS

df [–t] [–f] [file-systems]

DESCRIPTION

The *df* command prints out the number of free blocks and free i-nodes available for on-line file systems by examining the counts kept in the super-blocks; *file-systems* may be specified either by device name (e.g., **/dev/dsk/c1d0s2** or by mounted directory name (e.g., /usr). If the *file-systems* argument is unspecified, the free space on all of the mounted file systems is printed.

The *df* command uses the following options:

–t causes the total allocated block figures to be reported as well

–f only an actual count of the blocks in the free list is made (free i-nodes are not reported). With this option, *df* will report on raw devices.

FILES

/dev/dsk/*
/etc/mnttab

SEE ALSO

fs(4), mnttab(4) in the *AT&T 3B2 Computer Programmer Reference Manual.*

NAME

disks – adds missing /dev/entries for hard disks based on the Equipped Device Table (EDT)

SYNOPSIS

/etc/disks

DESCRIPTION

Disks will search the EDT to see which hard disks are equipped. For each equipped hard disk, the following steps are performed:

1. The **/dev/dsk** and **/dev/rdsk** directories are checked for an entry with the name **c[slot]d?s6**. Where **[slot]** is the slot the disk controller board is plugged into which is 0 for the disks controlled by the integral disk controller on the system board. The **?** is the number of the disk attached to the controller. The system board disk controller is capable of controlling two disks: 0 and 1.
2. If either entry is not found. Disks creates **/dev/dsk** and **/dev/rdsk** entries for the disk. The **/dev/SA** and **/dev/rSA** entries are created and linked to the **c[slot]d?s6** entry in **/dev/dsk** and **/dev/rdsk** respectively. The **/dev/SA** and **/dev/rSA** entries are named diskx where *x* is the lowest unused number for disk entries. A message is printed indicating that **/dev** files have been created.

Disks is called each time the system is booted. It must also be called after "sysadm rmdisk" to restore the **/dev** entries so the disk can be repartitioned.

FILES

/dev/dsk/*	entries for the hard disk for general use
/dev/rdsk	
/dev/SA	entries for the hard disk for use by Simple Administration
/dev/rSA	

SEE ALSO

sysadm(1) in the *AT&T 3B2 Computer User Reference Manual.*

NAME

drvinstall – install/uninstall a driver

SYNOPSIS

/etc/drvinstall [**–m** master] [**–d** object] [**–s** system] [**–o** directory] **–v** version **–nbufx**

DESCRIPTION

Drvinstall accepts an *object* file, *master* file, and *system* file as inputs and creates the corresponding specially formatted files for use by the self-configuring boot.

If the file is a software driver then *drvinstall* will assign the first available major number to that driver. This major number will be inserted into the *master* file. The major numbers available for software drivers on for 3B2 Computer 48-127 inclusive. The remaining major numbers are either reserved for hardware devices or used by integral drivers.

Drvinstall determines the available major numbers by scanning all existing *master* files for major numbers and then assigns the first major number from the above list.

Drvinstall will print the major number assigned, or the major number found in the *master* file if installing/uninstalling a software driver. For hardware drivers or loadable modules nothing is printed.

One or both of the **–m** or **–d** options must be specified. *Drvinstall* expects any unused field of the *master* file to be filled with a "–".

Drvinstall inserts/deletes **INCLUDE** statements in the *system* file for loadable object modules and software drivers.

The **–m** *master* argument specifies the path name of the *master* file to be used. If this flag is omitted, the **/etc/master.d** directory is used. The master file must reside in the root file system.

The **–d** *object* argument specifies the path name of the *object* file to be used. If this flag is omitted, the **/boot** directory is used.

The **–s** *system* argument specifies the path name of the *system* file to be used. If this flag is omitted, the **/etc/system** file is used.

The **–o** *directory* argument specifies the path name of the bootable file. If this flag is omitted, the **/boot** directory is used.

The **–v** *version* argument specifies the *version* number of *drvinstall* command compatible with the *master* file being used. This is a required option. The current version of driver installed is 1.0

The **–u** option will uninstall a driver. Either the **–d** or **–m** option must be used in conjunction with the **-u** option. A driver dependency check is made and if a dependency is found, a warning message is issused and the command is aborted. If no dependency is found, then the following actions are taken:

1. the bootable *object* file is removed.
2. the major number is replaced with a "–" in the *master* file if a software driver.
3. the **INCLUDE** statement is deleted from the *system* file.

The **–f** option, when used with the **–u** option, disables the dependency check. This will result in the driver being uninstalled.

The **–n** option inhibits any edit of the *system* file.

The −b option inhibits generation of the bootable *object* file.

The −x option enables debugging output.

SEE ALSO

mkboot(1M).
master(4), system(4) in the *AT&T 3B2 Computer Programmer Reference Manual.*

DIAGNOSTICS

The major numbered assigned or found for a software driver is printed on stdout. A zero is returned for success and an non-zero is returned for failures.

NAME

du – summarize disk usage

SYNOPSIS

du [**–sar**] [names]

DESCRIPTION

Du gives the number of blocks contained in all files and (recursively) directories within each directory and file specified by the *names* argument. The block count includes the indirect blocks of the file. If *names* is missing, . is used.

The optional arguments are as follows:

–s causes only the grand total (for each of the specified *names*) to be given.

–a causes an entry to be generated for each file.

Absence of either –s or –a causes an entry to be generated for each directory only.

–r will cause *du* to generate messages about directories that cannot be be read, files that cannot be opened, etc., rather than being silent (the default).

A file with two or more links is only counted once.

BUGS

If the –a option is not used, non-directories given as arguments are not listed.
If there are too many distinct linked files, *du* will count the excess files more than once.
Files with holes in them will get an incorrect block count.

NAME

errdump — print error log

SYNOPSIS

errdump

DESCRIPTION

This command displays on the system console the error log contained in the system's nonvolatile ram. The display contains the previous saved system state, the last 5 panic messages and their time of occurrence, and an indication of the log's sanity.

DIAGNOSTICS

The phrase "not superuser" if invoked by other than the superuser. Superuser is defined as individuals logged in under the root directory from the console port.

EXAMPLE

The following is an example of the printout in response to the *errdump* command.

```
#
#
#
#
# errdump
nvram status:    sane

csr:      0x0648   (floppy)   (unassigned)   (clock)   (uart)

psw:      rsvd CSH_F_D QIE CSH_D OE NZVC TE IPL
             0        1   0     1  0    0  0   f
           CM PM R I ISC TM FT
           0  0 1 0   5  0  3

r3:       0x00049001
r4:       0x00000081
r5:       0x00000000
r6:       0x40091348
r7:       0x0001a13f
r8:       0x4008edd8
oap:      0x400816d8
opc:      0x400083bc
osp:      0x40081700
ofp:      0x40081700
isp:      0x40080008
pcbp:     0x40041a40

fltar:    0xc0021140
fltcr:    reqacc   xlevel   ftype
          0xa      0x0      0x0

          srama             sramb
[0]       0x02034800        0x0000011f
[1]       0x02035100        0x00000030
[2]       0x02035860        0x00000074
[3]       0x02035c00        0x00000015
```

```
        Panic log

[0]     Thu Sep 20 09:51:36 1984
        KERNEL DATA ALIGNMENT ERROR

[1]     Thu Sep 20 09:51:37 1984
        KERNEL DATA ALIGNMENT ERROR

[2]     Thu Sep 20 09:51:40 1984
        KERNEL DATA ALIGNMENT ERROR

[3]     Thu Sep 20 09:52:21 1984
        KERNEL DATA ALIGNMENT ERROR

[4]     Fri Sep 21 05:50:10 1984
        SYSTEM PARITY ERROR INTERRUPT
```

SEE ALSO

UNIX system error messages in the *AT&T 3B2 Computer System Administration Reference Manual.*

NAME

ff − list file names and statistics for a file system

SYNOPSIS

/etc/ff [options] special

DESCRIPTION

Ff reads the i-list and directories of the *special* file, assuming it to be a file system, saving i-node data for files which match the selection criteria. Output consists of the path name for each saved i-node, plus any other file information requested using the print *options* below. Output fields are positional. The output is produced in i-node order; fields are separated by tabs. The default line produced by *ff* is:

path-name i-number

With all *options* enabled, output fields would be:

path-name i-number size uid

The argument *n* in the *option* descriptions that follow is used as a decimal integer (optionally signed), where +*n* means more than *n*, −*n* means less than *n*, and *n* means exactly *n*. A day is defined as a 24 hour period.

−**I**	Do not print the i-node number after each path name.
−**l**	Generate a supplementary list of all path names for multiply linked files.
−**p** *prefix*	The specified *prefix* will be added to each generated path name. The default is ..
−**s**	Print the file size, in bytes, after each path name.
−**u**	Print the owner's login name after each path name.
−**a** *n*	Select if the i-node has been accessed in *n* days.
−**m** *n*	Select if the i-node has been modified in *n* days.
−**c** *n*	Select if the i-node has been changed in *n* days.
−**n** *file*	Select if the i-node has been modified more recently than the argument *file*.
−**i** *i-node-list*	Generate names for only those i-nodes specified in *i-node-list*.

SEE ALSO

ncheck(1M).
find(1) in the *AT&T 3B2 Computer User Reference Manual.*

BUGS

Only a single path name out of any possible ones will be generated for a multiply linked i-node, unless the −**I** option is specified. When −**l** is specified, no selection criteria apply to the names generated. All possible names for every linked file on the file system will be included in the output.

On very large file systems, memory may run out before *ff* does.

NAME

finc – fast incremental backup

SYNOPSIS

finc [selection-criteria] file-system raw-tape

DESCRIPTION

Finc selectively copies the input *file-system* to the output *raw-tape* . The cautious will want to mount the input *file-system* read-only to insure an accurate backup, although acceptable results can be obtained in read-write mode. The tape must be previously labelled by *labelit*. The selection is controlled by the *selection-criteria*, accepting only those inodes/files for whom the conditions are true.

It is recommended that production of a *finc* tape be preceded by the *ff* command, and the output of *ff* be saved as an index of the tape's contents. Files on a *finc* tape may be recovered with the *frec* command.

The argument *n* in the *selection-criteria* which follow is used as a decimal integer (optionally signed), where *+n* means more than *n*, *−n* means less than *n*, and *n* means exactly *n*. A day is defined as a 24 hours.

−a *n*	True if the file has been accessed in *n* days.
−m *n*	True if the file has been modified in *n* days.
−c *n*	True if the i-node has been changed in *n* days.
−n *file*	True for any file which has been modified more recently than the argument *file*.

EXAMPLES

To write a tape consisting of all files from file-system **/usr** modified in the last 48 hours:

```
finc −m −2 /dev/rdsk/c1d0s2 /dev/rSA/ctape1
```

SEE ALSO

ff(1M), frec(1M), labelit(1M).
cpio(1) in the *AT&T 3B2 Computer User Reference Manual*.

NAME

fmtflop – physically format floppy disks

SYNOPSIS

fmtflop [**-v**] *device*

DESCRIPTION

Fmtflop physically formats the 96 tpi floppy media inserted in the floppy disk drive. The **-v** option formats and verifies that the formatted floppy disk is correct. The *device* is the path name of the floppy disk drive (e.g., **/dev/idsk/c0d0s6**).

Fmtflop formats DOUBLE SIDED media with 512 byte sectors, 9 sectors per track, and 80 tracks. Before executing *fmtflop*, the floppy media must be placed in the disk drve and the door must be closed.

SEE ALSO

if(7).

DIAGNOSTICS

An error message is returned if the format or verify fails. If this occurs, it is best to reissue the command. If the command fails the second time (especially on the same area of the disk) the floppy media is probably bad.

NAME

fmthard – populate VTOC on hard disks

SYNOPSIS

fmthard [–i] [–m] [–s datafile] [–q] [–n volumename] [–v] /dev/rdsk/c?d?s6

DESCRIPTION

Fmthard populates the VTOC (volume table of contents) on the **hard** disks.

The following options apply to fmthard:

–i just prints the default file name which would be used to set up the VTOC.

–m automatically makes a file system on each mountable partition.

–s the VTOC is populated according to a *datafile* created by the user. The *datafile* format is described below.

–q checks disk for older format (non-VTOC). returns 0 for older format; otherwise a 1 is returned.

–n allows the disk to be given a volume name up to 8 character long.

–v verbose mode in which progress of the command is returned to screen. When used with the –m option, parameters of file systems are returned.

If no options are given, the VTOC is populated according to the specification of a default file, */etc/vtoc*, where *x* is the drive id of the disk. For example, for the 30M disk, the default file is labeled */etc/vtoc/hd3dft*.

The *datafile* contains one line for each partition, listed in consecutive order. Each line is delimited by a newline. Comment lines may be inserted, the first character being an asterisk. Each line is composed of entries that are position dependent separated by white space and have the following format:

partition id flag Start Sector Size in Sectors

Where the entries have the following values.

partition The partition number 0-15.

id The partition id consists of a two digit hex number. The following being reserved codes 00 unassigned, V_BOOT 0x01, V_ROOT 0x02, V_SWAP 0x03, V_USR 0x04 and V_BACKUP 0x05. (id is called TAG in prtvtoc)

flag The flag allows a partition to be flagged as unmountable or read only, the masks being: 00_mount, read/write, V_UNMNT 0x01, and V_RONLY 0x10.

Start Sector Start Sector is defined as the sector number on which the partition starts.

Size in Sectors The size of the partition is specified by the number of sectors (512 byte blocks).

FILES

/etc/vtoc/*

NAME

frec – recover files from a backup tape

SYNOPSIS

/etc/frec [**–p** path] [**–f** reqfile] raw-tape i-number:name ...

DESCRIPTION

Frec recovers files from the specified *raw-tape* backup tape written by *finc*(1M), given their *i-numbers*. The data for each recovery request will be written into the file given by *name* .

The **–p** option allows you to specify a default prefixing *path* different from your current working directory. This will be prefixed to any *names* that are not fully qualified, i.e. that do not begin with / or ./. If any directories are missing in the paths of recovery *names* they will be created.

–p *path* Specifies a prefixing *path* to be used to fully qualify any names that do not start with / or ./.

–f *reqfile* Specifies a file which contains recovery requests. The format is i-number:newname, one per line.

EXAMPLES

To recover a file, i-number 1216 when backed-up, into a file named *junk* in your current working directory:

```
frec /dev/rSA/ctape1 1216:junk
```

To recover files with i-numbers 14156, 1232, and 3141 into files **/usr/src/cmd/a**, **/usr/src/cmd/b** and **/usr/joe/a.c**:

```
frec –p /usr/src/cmd /dev/rSA/ctape1 14156:a 1232:b
3141:/usr/joe/a.c
```

SEE ALSO

ff(1M), finc(1M), labelit(1M).
cpio(1) in the *AT&T 3B2 Computer User Reference Manual.*

BUGS

While paving a path (i.e. creating the intermediate directories contained in a pathname) *frec* can only recover inode fields for those directories contained on the tape and requested for recovery.

NAME

fsck, dfsck − file system consistency check and interactive repair

SYNOPSIS

/etc/fsck [−**y**] [−**n**] [−**sX**] [−**SX**] [−**t** file] [−**q**] [−**D**] [−**f**] [−**b**] [file-systems]
/etc/dfsck [options1] filsys1 ... − [options2] filsys2 ...

DESCRIPTION

Fsck

Fsck audits and interactively repairs inconsistent conditions for UNIX system files. If the file system is consistent then the number of files, number of blocks used, and number of blocks free are reported. If the file system is inconsistent the user is prompted for concurrence before each correction is attempted. It should be noted that most corrective actions will result in some loss of data. The amount and severity of data lost may be determined from the diagnostic output. The default action for each consistency correction is to wait for the user to respond **yes** or **no**. If the user does not have write permission *fsck* will default to a −**n** action.

Fsck has more consistency checks than its predecessors *check*, *dcheck*, *fcheck*, and *icheck* combined.

The following options are interpreted by *fsck*.

−**y** Assume a **yes** response to all questions asked by *fsck*.

−**n** Assume a **no** response to all questions asked by *fsck*; do not open the file system for writing.

−**s***X* Ignore the actual free list and (unconditionally) reconstruct a new one by rewriting the super-block of the file system. The file system should be unmounted while this is done; if this is not possible, care should be taken that the system is quiescent and that it is rebooted immediately afterwards. This precaution is necessary so that the old, bad, in-core copy of the superblock will not continue to be used, or written on the file system.

X=blks/cyl:rotational gap
=72:9 10M hard disk
=90:9 32M hard disk
=18:1 floppy disk

The −s*X* option allows for creating an optimal free-list organization.

If *X* is not given, the values used when the file system was created are used.

−**S***X* Conditionally reconstruct the free list. This option is like −**s***X* above except that the free list is rebuilt only if there were no discrepancies discovered in the file system. Using −**S** will force a **no** response to all questions asked by *fsck*. This option is useful for forcing free list reorganization on uncontaminated file systems.

−**t** If *fsck* cannot obtain enough memory to keep its tables, it uses a scratch file. If the −**t** option is specified, the file named in the next argument is used as the scratch file, if needed. Without the −**t** flag, *fsck* will prompt the user for the name of the scratch file. The file chosen should not be on the file system being checked, and if it is not a special file or did not already exist, it is removed when *fsck* completes.

−**q** Quiet *fsck*. Do not print size-check messages. Unreferenced **fifos** will silently be removed. If *fsck* requires it, counts in the superblock will be automatically fixed and the free list salvaged.

-D Directories are checked for bad blocks. Useful after system crashes.

-f Fast check. Check block and sizes and check the free list. The free list will be reconstructed if it is necessary.

-b Reboot. If the file system being checked is the root file system and modifications have been made, then either remount the root file system or reboot the system. A remount is done only if there was minor damage.

If no *file-systems* are specified, *fsck* will read a list of default file systems from the file **/etc/checklist**.

Inconsistencies checked are as follows:

1. Blocks claimed by more than one i-node or the free list.
2. Blocks claimed by an i-node or the free list outside the range of the file system.
3. Incorrect link counts.
4. Size checks:
   Incorrect number of blocks.
   Directory size not 16-byte aligned.
5. Bad i-node format.
6. Blocks not accounted for anywhere.
7. Directory checks:
   File pointing to unallocated i-node.
   I-node number out of range.
8. Super Block checks:
   More than 65536 i-nodes.
   More blocks for i-nodes than there are in the file system.
9. Bad free block list format.
10. Total free block and/or free i-node count incorrect.

Orphaned files and directories (allocated but unreferenced) are, with the user's concurrence, reconnected by placing them in the **lost+found** directory, if the files are nonempty. The user will be notified if the file or directory is empty or not. If it is empty, *fsck* will silently remove them. *Fsck* will force the reconnection of nonempty directories. The name assigned is the i-node number. The only restriction is that the directory **lost+found** must preexist in the root of the file system being checked and must have empty slots in which entries can be made. This is accomplished by making **lost+found**, copying a number of files to the directory, and then removing them (before *fsck* is executed).

Checking the raw device is almost always faster and should be used with everything but the *root* file system.

Dfsck

Dfsck allows two file system checks on two different drives simultaneously. *options1* and *options2* are used to pass options to *fsck* for the two sets of file systems. A **-** is the separator between the file system groups.

The *dfsck* program permits a user to interact with two *fsck* programs at once. To aid in this, *dfsck* will print the file system name for each message to the user. When answering a question from *dfsck*, the user must prefix the response with a **1** or a **2** (indicating that the answer refers to the first or second file system group). The following serves as an example:

/etc/dfsck /dev/dsk/c1d0s6 /dev/dsk/c1d0s5

WARNINGS

Do not use *dfsck* to check the *root* file system.

FILES

/etc/checklist	contains default list of file systems to check.
/etc/checkall	optimizing *dfsck* shell file.

SEE ALSO

checkall(1M), mkfs(1M), ncheck(1M), crash(1M).
uadmin(2), checklist(4), fs(4) in the *AT&T 3B2 Computer Programmer Reference Manual.*

BUGS

I-node numbers for . and .. in each directory should be checked for validity.

DIAGNOSTICS

The diagnostics produced by *fsck* are intended to be self-explanatory.

NAME

fsdb – file system debugger

SYNOPSIS

/etc/fsdb special [–]

DESCRIPTION

Fsdb can be used to patch up a damaged file system after a crash. It has conversions to translate block and i-numbers into their corresponding disk addresses. Also included are mnemonic offsets to access different parts of an i-node. These greatly simplify the process of correcting control block entries or descending the file system tree.

Fsdb contains several error-checking routines to verify i-node and block addresses. These can be disabled if necessary by invoking *fsdb* with the optional – argument or by the use of the **O** symbol. (*Fsdb* reads the i-size and f-size entries from the superblock of the file system as the basis for these checks.)

Numbers are considered decimal by default. Octal numbers must be prefixed with a zero. During any assignment operation, numbers are checked for a possible truncation error due to a size mismatch between source and destination.

Fsdb reads a block at a time and will therefore work with raw as well as block I/O. A buffer management routine is used to retain commonly used blocks of data in order to reduce the number of read system calls. All assignment operations result in an immediate write-through of the corresponding block.

The symbols recognized by *fsdb* are:

#	absolute address
i	convert from i-number to i-node address
b	convert to block address
d	directory slot offset
+, –	address arithmetic
q	quit
>, <	save, restore an address
=	numerical assignment
=+	incremental assignment
=–	decremental assignment
="	character string assignment
O	error checking flip flop
p	general print facilities
f	file print facility
B	byte mode
W	word mode
D	double word mode
!	escape to shell

The print facilities generate a formatted output in various styles. The current address is normalized to an appropriate boundary before printing begins. It advances with the printing and is left at the address of the last item printed. The output can be terminated at any time by typing the delete character. If a number follows the **p** symbol, that many entries are printed. A check is made to detect block boundary overflows since logically sequential blocks are generally not physically sequential. If a count of zero is used, all entries to the end of the current block are printed.

The print options available are:

i	print as i-nodes
d	print as directories
o	print as octal words
e	print as decimal words
c	print as characters
b	print as octal bytes

The **f** symbol is used to print data blocks associated with the current i-node. If followed by a number, that block of the file is printed. (Blocks are numbered from zero.) The desired print option letter follows the block number, if present, or the **f** symbol. This print facility works for small as well as large files. It checks for special devices and that the block pointers used to find the data are not zero.

Dots, tabs, and spaces may be used as function delimiters but are not necessary. A line with just a new-line character will increment the current address by the size of the data type last printed. That is, the address is set to the next byte, word, double word, directory entry or i-node, allowing the user to step through a region of a file system. Information is printed in a format appropriate to the data type. Bytes, words and double words are displayed with the octal address followed by the value in octal and decimal. A **.B** or **.D** is appended to the address for byte and double word values, respectively. Directories are printed as a directory slot offset followed by the decimal i-number and the character representation of the entry name. I-nodes are printed with labeled fields describing each element.

The following mnemonics are used for i-node examination and refer to the current working i-node:

md	mode
ln	link count
uid	user ID number
gid	group ID number
sz	file size
a#	data block numbers (0 − 12)
at	access time
mt	modification time
maj	major device number
min	minor device number

EXAMPLES

386i	prints i-number 386 in an i-node format. This now becomes the current working i-node.
ln=4	changes the link count for the working i-node to 4.
ln=+1	increments the link count by 1.
fc	prints, in ASCII, block zero of the file associated with the working i-node.
2i.fd	prints the first 32 directory entries for the root i-node of this file system.
d5i.fc	changes the current i-node to that associated with the 5th directory entry (numbered from zero) found from the above command. The first logical block of the file is then printed in ASCII.
512B.p0o	prints the superblock of this file system in octal.

2i.a0b.d7=3	changes the i-number for the seventh directory slot in the root directory to 3. This example also shows how several operations can be combined on one command line.
d7.nm="name"	changes the name field in the directory slot to the given string. Quotes are optional when used with **nm** if the first character is alphabetic.
a2b.p0d	prints the third block of the current i-node as directory entries.

SEE ALSO

fsck(1M).

dir(4), fs(4) in the *AT&T 3B2 Computer Programmer Reference Manual.*

NAME

fsstat – file system status

SYNOPSIS

/etc/fsstat file-system

DESCRIPTION

Fsstat reports on the status of *file-system*. It succeeds if the file system is unmounted and appears okay. For the root file system, it succeeds if it is active and not marked bad.

SEE ALSO

fs(4) in the *AT&T 3B2 Computer Programmer Reference Manual.*

DIAGNOSTICS

If successful, the command has an exit status of 0. Otherwise, the command has an exit status of 1 if the file system needs to be checked, 2 if mounted, and 3 for other failures.

NAME

fuser – identify processes using a file or file structure

SYNOPSIS

/etc/fuser –[**ku**] [file 1 . . .]

DESCRIPTION

Fuser lists the process IDs of the processes using the *files* specified as arguments. For block special devices, all processes using any file on that device are listed. The process ID is followed by **c**, **p** or **r** if the process is using the file as its current directory, the parent of its current directory (only when in use by the system), or its root directory, respectively.

The following options may be used with *fuser*:

–**u** the login name, in parentheses, also follows the process ID.

–**k** the SIGKILL signal is sent to each process. Only the super-user can terminate another user's process [see *kill*(2)].

Options may be respecified between groups of files. The new set of options replaces the old set, with a lone dash canceling any options currently in force.

The process IDs are printed as a single line on the standard output, separated by spaces and terminated with a single new line. All other output is written on standard error.

FILES

/unix	for namelist
/dev/kmem	for system image
/dev/mem	also for system image

SEE ALSO

mount(1M).
ps(1) in the *AT&T 3B2 Computer User Reference Manual.*
kill(2), signal(2) in the *AT&T 3B2 Computer Programmer Reference Manual.*

NAME

getmajor – print slot/major number(s) of hardware devices

SYNOPSIS

/etc/getmajor name | bdcode

DESCRIPTION

Getmajor prints all slot/major numbers found in system equipped device table for the requested device. The argument *name* may be entered in lower or upper case.

DIAGNOSTICS

If successful a zero is returned. If *name* | *bdcode* is not found, a NULL is printed and a nonzero is returned.

NAME

getty − set terminal type, modes, speed, and line discipline

SYNOPSIS

/etc/getty [**−h**] [**−t** timeout] line [speed [type [linedisc]]]
/etc/getty **−c** file

DESCRIPTION

Getty is a program that is invoked by *init*(1M). It is the second process in the series, (*init-getty-login-shell*) that ultimately connects a user with the UNIX system. Initially *getty* prints the login message field for the entry it is using from **/etc/gettydefs**. *Getty* reads the user's login name and invokes the *login*(1) command with the user's name as argument. While reading the name, *getty* attempts to adapt the system to the speed and type of terminal being used.

Line is the name of a tty line in **/dev** to which *getty* is to attach itself. *Getty* uses this string as the name of a file in the **/dev** directory to open for reading and writing. Unless *getty* is invoked with the **−h** flag, *getty* will force a hangup on the line by setting the speed to zero before setting the speed to the default or specified speed. The **−t** flag plus *timeout* in seconds, specifies that *getty* should exit if the open on the line succeeds and no one types anything in the specified number of seconds. The optional second argument, *speed*, is a label to a speed and tty definition in the file **/etc/gettydefs**. This definition tells *getty* at what speed to initially run, what the login message should look like, what the initial tty settings are, and what speed to try next should the user indicate that the speed is inappropriate (by typing a <*break*> character). The default *speed* is 300 baud. The optional third argument, *type*, is a character string describing to *getty* what type of terminal is connected to the line in question. *Getty* understands the following types:

none	default
vt61	DEC vt61
vt100	DEC vt100
hp45	Hewlett-Packard 45
c100	Concept 100

The default terminal is **none**; i.e., any crt or normal terminal unknown to the system. Also, for terminal type to have any meaning, the virtual terminal handlers must be compiled into the operating system. They are available, but not compiled in the default condition. The optional fourth argument, *linedisc*, is a character string describing which line discipline to use in communicating with the terminal. Again the hooks for line disciplines are available in the operating system but there is only one presently available, the default line discipline, **LDISC0**.

When given no optional arguments, *getty* sets the *speed* of the interface to 300 baud, specifies that raw mode is to be used (awaken on every character), that echo is to be suppressed, either parity allowed, new-line characters will be converted to carriage return-line feed, and tab expansion performed on the standard output. It types the login message before reading the user's name a character at a time. If a null character (or framing error) is received, it is assumed to be the result of the user pushing the "break" key. This will cause *getty* to attempt the next *speed* in the series. The series that *getty* tries is determined by what it finds in **/etc/gettydefs**.

The user's name is terminated by a new-line or carriage-return character. The latter results in the system being set to treat carriage returns appropriately (see *ioctl*(2)).

The user's name is scanned to see if it contains any lower-case alphabetic characters; if not, and if the name is non-empty, the system is told to map any future upper-case characters into the corresponding lower-case characters.

Finally, *login* is called with the user's name as an argument. Additional arguments may be typed after the login name. These are passed to *login*, which will place them in the environment (see *login*(1)).

A check option is provided. When *getty* is invoked with the −c option and *file*, it scans the file as if it were scanning **/etc/gettydefs** and prints out the results to the standard output. If there are any unrecognized modes or improperly constructed entries, it reports these. If the entries are correct, it prints out the values of the various flags. See *ioctl*(2) to interpret the values. Note that some values are added to the flags automatically.

FILES

/etc/gettydefs
/etc/issue

SEE ALSO

init(1M), tty(7).
ct(1C), login(1) in the *AT&T 3B2 Computer User Reference Manual.*
ioctl(2), gettydefs(4), inittab(4) in the *AT&T 3B2 Computer Programmer Reference Manual.*

BUGS

While *getty* does understand simple single character quoting conventions, it is not possible to quote the special control characters that *getty* uses to determine when the end of the line has been reached, which protocol is being used, and what the erase character is. Therefore it is not possible to login via *getty* and type a #, @, /, !, _, backspace, ^U, ^D, or & as part of your login name or arguments. They will always be interpreted as having their special meaning as described above.

NAME

hdeadd – add/delete reports to/from Hard Disk Error (HDE) Log

SYNOPSIS

hdeadd **–a** [*aoptions*]

hdeadd **–d** [*doptions*]

hdeadd **–e** [[**–D**] major minor]

hdeadd **–f** filename

hdeadd **–r** [**–D**] major minor filename

hdeadd **–s** [**–D**] major minor filename

DESCRIPTION

This command is a bad block handling utility command. **You must be super-user to use it.** It is used to print the list of equipped disks. It is also used for manually adding or deleting disk error reports. The manual mechanism is used only when the system has reported a bad block and was in an inappropriate state for manually logging it. These include disk errors reported while in firmware mode and disk errors that cause the system to PANIC. In addition, this command has some options that are intended to be used only while testing the feature.

The following options may be used with *hdeadd*:

–a *hdeadd* allows a Hard Disk Error (HDE) report to be manually added to the HDE Log of a disk.

–d *hdeadd* allows a specific report or a range of reports to be deleted from the HDE Log of a disk.

–e prints out the list of major/minor device numbers of the equipped hard disks. If the *major* and *minor* device numbers are also provided, it determines if that specification is an equipped hard disk. The result is both printed on the standard output and is used to determine the exit status. A NORMAL (or TRUE) exit means it is an equipped disk.

–f the file specified by *filename* is assumed to contain a canned set of HDE Log manipulations. Each line of text contains one specification in the command argument form, starting with a **–a** or a option.

–s saves a copy of the HDE Log of the specified (by *major/minor* device number) disk in the file specified by *filename*.

–r restores the HDE Log of the specified disk from the file specified by *filename*.

The valid *aoptions* are *hard disk error* specifications.

The valid *doptions* are either a *hard disk error* specification or an *error range* specification.

A *hard disk error* specification includes the following values:

–D *maj min* Specifies the major device number (*maj*) and minor device number (*min*) of the disk.

–b *blockno* **Normal form:** Specifies the physical disk block number in integer counter form (e.g., treating the disk as a simple stream of blocks). Physical disk block numbering starts with zero meaning sector 0 of track 0 of cylinder 0. This is the normal form that is reported by the

operating system.

−B *cyl trk sec* **Alternate form:** Specifies the physical disk block number in in terms of its physical cylinder number (*cyl*), track number within cylinder (*trk*), and sector number within track (*sec*). This alternate form is available to cover the possibility of a non-operating system detector reporting block numbers in this hardware form.

−t *mmddhhmm*[*yy*] **Optional:** Specifies the time of day when the error actually occurred. If omitted when adding reports, the current time is used. If omitted when deleting reports, any reports for the given block are deleted.

An *error range* specification includes the following values:

−D *maj min* Specifies the major device number (*maj*) and minor device number (*min*) of the disk.

−F *mmddhhmm*[*yy*] **Optional:** Specifies the "from" time for the time interval being purged. If omitted, zero (the beginning of time) is used.

−T *mmddhhmm*[*yy*] **Optional:** Specifies the "to" time for the time interval being purged. If omitted, the end of time is used. The range comparisons include the end values of the range in the purge.

FILES

/dev/hdelog

SEE ALSO

hdefix(1M), hdelogger(1M), hdelog(7).

DIAGNOSTICS

The HDE commands exit with one of three values:

0 means NORMAL, or TRUE

1 means bad command usage or execution errors

2 means BAD BLOCKS or FALSE (but command executed successfully)

WARNINGS

You must be super-user to use *hdeadd*.

NAME

hdefix – report or change bad block mapping

SYNOPSIS

hdefix **-p** [[**-D**] major minor]

hdefix **-a** [major minor [*blocknospec* ...]]

hdefix **-F** [**-D**] major minor [*blocknospec* ...]]

hdefix **-r** [**-D**] major minor filename

hdefix **-s** [**-D**] major minor filename

DESCRIPTION

This command is a bad block handling utility command. You must be super-user to use it. The *hdefix* command is used to find out what blocks are currently mapped to surrogate images on the equipped hard disks and it is used the change what blocks are mapped. In addition, this command has some options that are intended to be used only while testing the feature.

When the mapping is changed, block initialization is performed. The original block is assumed to be unreadable and the new surrogate image is zeroed. Data is probably lost, so damage is expected.

If the block is associated with a file system, the file system may be damaged as a result of the mapping change. To handle this situation, the file system is marked dirty, which means *fsck*(1M) must be run before the file system can be used, and a system reboot is forced after all other bad block processing is complete. If the block is a data block of a file, that file will be corrupted, even after this recovery has finished.

The following options may be used with *hdefix*:

-p *hdefix* prints a report of media properites that includes the currently mapped bad blocks. If a specific disk is specified (by giving its *major* and *minor* device numbers), only the report for that disk is printed. If no disk is specified, a report is given for each equipped disk.

-D used to specify the major device number (*maj*) and minor device number (*min*) of the disk.

-a To map new bad blocks, the **-a** option is used. If no arguments follow the **-a** option, each equipped disk is processed, using the HDE Log on each disk to determine which blocks to map. If a specific disk is specified, only that disk is processed. If one or more block numbers are specified, those blocks are mapped, instead of using the HDE Log to get blocks to map. This is the only way to map an unreadable block containing the HDE Log.

-F forces blocks to be removed from the map without any attempt to initialize them. This is intended only for testing the Bad Block Handling feature. If no block number is specified, the last block in the map is removed and the block number is ouput on the standard output.

-s a copy of the bad block map table and the surrogates pointed to by the map is saved in the file specified by the *filename* argument.

-r the bad block map table and the surrogates pointed to by the map are restored from the file specified by the *filename* argument. The save and restore options are intended only for testing. The restore can be quite destructive unless used under very controlled conditions.

A *blocknospec* has the following forms:

−b *blockno*	Specifies the physical disk block number in integer counter form (e.g., treating the disk as a simple stream of blocks). Physical disk block numbering starts with zero meaning sector 0 of track 0 of cylinder 0.
−B *cyl trk sec*	Specifies the physical disk block number in in terms of its physical cylinder number (*cyl*), track number within cylinder (*trk*), and sector number within track (*sec*). Only one of the alternate forms of block number should be specified for a given block.

FILES

/dev/hdelog

SEE ALSO

hdeadd(1M), hdelogger(1M), hdelog(7).

DIAGNOSTICS

The HDE commands exit with one of three values:

0	means NORMAL, or TRUE
1	means bad command usage or execution errors
2	means BAD BLOCKS or FALSE (but command executed successfully)

WARNINGS

You must be super-user to use *hdefix*.

NAME

hdelogger – Hard Disk Error (HDE) status report command and Log Demon

SYNOPSIS

hdelogger [–s] [–f] [–D maj min]

DESCRIPTION

This command is a bad block handling utility command. **You must be super-user to use it.** The *hdelogger* command serves two purposes. When run by the **init** process (process 1 – see *init*(1M)), this command performs the functions of the Hard Disk Error (HDE) Log Demon. These functions include providing summaries of outstanding errors during system startup and shutdown transitions, along with adding new errors to HDE Logs and giving the revised status summaries as errors are reported by hard disk drivers. When run as the demon, no options are used.

When run as a normal command (process 1 is not its parent), this command provides on the spot reports of outstanding errors as recorded in the HDE Logs of equipped hard disks. The following options control report generation:

–s	Specifies that summary reports are to be generated. The summary report provides sufficient information for normal Bad Block Handling operations. This is the default.
–f	Specifies that full reports are to be generated. This is intended mainly for testing the Bad Block Handling feature, but is available in case maintenance personnel need additional detail for troubleshooting complicated problems.
–D *maj min*	Restricts the report generation to a specific hard disk. If this option is omitted, reports will be generated for all equipped hard disks.

FILES

/dev/hdelog

SEE ALSO

hdeadd(1M), hdefix(1M), hdelog(7).

DIAGNOSTICS

The HDE commands exit with one of three values:

0	means NORMAL, or TRUE
1	means bad command usage or execution errors
2	means BAD BLOCKS or FALSE (but command executed successfully)

WARNINGS

You must be super-user to use *hdelogger*.

NAME

id – print user and group IDs and names

SYNOPSIS

id

DESCRIPTION

Id writes a message on the standard output giving the user and group IDs and the corresponding names of the invoking process. If the effective and real IDs do not match, both are printed.

SEE ALSO

logname(1) in the *AT&T 3B2 Computer User Reference Manual.*
getuid(2) in the *AT&T 3B2 Computer Programmer Reference Manual.*

NAME

init, telinit – process control initialization

SYNOPSIS

/etc/init [0123456SsQq]

/etc/telinit [0123456sSQqabc]

DESCRIPTION

Init

Init is a general process spawner. Its primary role is to create processes from a script stored in the file **/etc/inittab** (see *inittab*(4)). This file usually has *init* spawn *getty*'s on each line that a user may log in on. It also controls autonomous processes required by any particular system.

Init considers the system to be in a *run-level* at any given time. A *run-level* can be viewed as a software configuration of the system where each configuration allows only a selected group of processes to exist. The processes spawned by *init* for each of these *run-levels* is defined in the *inittab* file. *Init* can be in one of eight *run-levels*, **0**–**6** and S or s. The *run-level* is changed by having a privileged user run **/etc/init** (which is linked to */etc/telinit*). This user-spawned *init* sends appropriate signals to the orginal *init* spawned by the operating system when the system was rebooted, telling it which *run-level* to change to.

Init is invoked inside the UNIX system as the last step in the boot procedure. The first thing *init* does is to look for **/etc/inittab** and see if there is an entry of the type *initdefault* (see *inittab*(4)). If there is, *init* uses the *run-level* specified in that entry as the initial *run-level* to enter. If this entry is not in *inittab* or *inittab* is not found, *init* requests that the user enter a *run-level* from the virtual system console, **/dev/console**. If an S (s) is entered, *init* goes into the *SINGLE USER* level. This is the only *run-level* that doesn't require the existence of a properly formatted *inittab* file. If **/etc/inittab** doesn't exist, then by default the only legal *run-level* that *init* can enter is the *SINGLE USER* level. In the *SINGLE USER* level the virtual console terminal **/dev/console** is opened for reading and writing and the command **/bin/su** is invoked immediately. To exit from the *SINGLE USER run-level* one of two options can be elected. First, if the shell is terminated (via an end-of-file), *init* will reprompt for a new *run-level*. Second, the *init* or *telinit* command can signal *init* and force it to change the *run-level* of the system.

When attempting to boot the system, failure of *init* to prompt for a new *run-level* may be due to the fact that the device **/dev/console** is linked to a device other than the physical system teletype (**/dev/contty**). If this occurs, *init* can be forced to relink **/dev/console** by typing a delete on the system teletype which is collocated with the processor.

When *init* prompts for the new *run-level*, the operator may enter only one of the digits **0** through **6** or the letters S or s. If S is entered *init* operates as previously described in *SINGLE USER* mode with the additional result that **/dev/console** is linked to the user's terminal line, thus making it the virtual system console. A message is generated on the physical console, **/dev/contty**, saying where the virtual terminal has been relocated.

When *init* comes up initially and whenever it switches out of *SINGLE USER* state to normal run states, it sets the *ioctl*(2) states of the virtual console, **/dev/console**, to those modes saved in the file **/etc/ioctl.console**. This file is written by *init* whenever *SINGLE USER* mode is entered. If this file does not exist when *init* wants to read it, a warning is printed and default settings are assumed.

If a **0** through **6** is entered *init* enters the corresponding *run-level*. Any other input will be rejected and the user will be re-prompted. If this is the first time *init* has entered a *run-level* other than *SINGLE USER*, *init* first scans *inittab* for special entries of the type *boot* and *bootwait*. These entries are performed, providing the *run-level* entered matches that of the entry before any normal processing of *inittab* takes place. In this way any special initialization of the operating system, such as mounting file systems, can take place before users are allowed onto the system. The *inittab* file is scanned to find all entries that are to be processed for that *run-level*.

Run-level **2** is usually defined by the user to contain all of the terminal processes and daemons that are spawned in the multi-user environment.

In a multi-user environment, the *inittab* file is usually set up so that *init* will create a process for each terminal on the system.

For terminal processes, ultimately the shell will terminate because of an end-of-file either typed explicitly or generated as the result of hanging up. When *init* receives a child death signal, telling it that a process it spawned has died, it records the fact and the reason it died in **/etc/utmp** and **/etc/wtmp** if it exists (see *who*(1)). A history of the processes spawned is kept in **/etc/wtmp** if such a file exists.

To spawn each process in the *inittab* file, *init* reads each entry and for each entry which should be respawned, it forks a child process. After it has spawned all of the processes specified by the *inittab* file, *init* waits for one of its descendant processes to die, a powerfail signal, or until *init* is signaled by *init* or *telinit* to change the system's *run-level*. When one of the above three conditions occurs, *init* re-examines the *inittab* file. New entries can be added to the *inittab* file at any time; however, *init* still waits for one of the above three conditions to occur. To provide for an instantaneous response the **init Q** or **init q** command can wake *init* to re-examine the *inittab* file.

If *init* receives a *powerfail* signal (*SIGPWR*) and is not in *SINGLE USER* mode, it scans *inittab* for special powerfail entries. These entries are invoked (if the *run-levels* permit) before any further processing takes place. In this way *init* can perform various cleanup and recording functions whenever the operating system experiences a power failure.

When *init* is requested to change *run-levels* (via *telinit*), *init* sends the warning signal (SIGTERM) to all processes that are undefined in the target *run-level*. *Init* waits 20 seconds before forcibly terminating these processes via the kill signal (SIGKILL).

Telinit

Telinit, which is linked to */etc/init*, is used to direct the actions of *init*. It takes a one-character argument and signals *init* via the *kill* system call to perform the appropriate action. The following arguments serve as directives to *init*.

0–6	tells *init* to place the system in one of the *run-levels* **0–6**.
a,b,c	tells *init* to process only those **/etc/inittab** file entries having the **a**, **b** or **c** *run-level* set.
Q,q	tells *init* to re-examine the **/etc/inittab** file.
s,S	tells *init* to enter the single user environment. When this level change is effected, the virtual system teletype, **/dev/console**, is changed to the terminal from which the command was executed.

FILES

/etc/inittab

/etc/utmp
/etc/wtmp
/etc/ioctl.console
/dev/console
/dev/contty

SEE ALSO

getty(1M).
login(1), sh(1), who(1) in the *AT&T 3B2 Computer User Reference Manual.*
kill(2), inittab(4), utmp(4) in the *AT&T 3B2 Computer Programmer Reference Manual.*

DIAGNOSTICS

If *init* finds that it is continuously respawning an entry from **/etc/inittab** more than 10 times in 2 minutes, it will assume that there is an error in the command string, and generate an error message on the system console, and refuse to respawn this entry until either 5 minutes has elapsed or it receives a signal from a user *init* (*telinit*). This prevents *init* from eating up system resources when someone makes a typographical error in the *inittab* file or a program is removed that is referenced in the *inittab*.

WARNINGS

Telinit can only be run by someone who is super-user or a member of group sys.

NAME

killall – kill all active processes

SYNOPSIS

/etc/killall [signal]

DESCRIPTION

Killall is a procedure used by **/etc/shutdown** to kill all active processes not directly related to the shutdown procedure.

Killall is chiefly used to terminate all processes with open files so that the mounted file systems will be unbusied and can be unmounted.

Killall sends *signal* (see *kill*(1)) to all remaining processes not belonging to the above group of exclusions. If no *signal* is specified, a default of **9** is used.

FILES

/etc/shutdown

SEE ALSO

fuser(1M), shutdown(1M).
kill(1), ps(1) in the *AT&T 3B2 Computer User Reference Manual.*
signal(2) in the *AT&T 3B2 Computer Programmer Reference Manual.*

WARNINGS

The *killall* command can only be executed by the super-user.

NAME

labelit − provide labels for disk file systems

SYNOPSIS

/etc/labelit special [fsname volume [**−n**]]

DESCRIPTION

Labelit is used to provide initial labels for unmounted file systems. With the optional arguments omitted, *labelit* prints current label values for mounted or unmounted file systems of the specified partition. The **−n** option provides for initial labeling only (this destroys previous contents).

The *fsname* argument represents the mounted name (e.g.,: **root**, **u1**, etc.) of the filesystem.

The *special* should be the physical disk section (e.g.,: **/dev/(r)dsk/c0d0s6**).

Fsname and *volumename* are recorded in the last 12 characters of the superblock (**char fsname[6], volname[6];**).

SEE ALSO

sh(1) in the *AT&T 3B2 Computer User Reference Manual.*
fs(4) in the *AT&T 3B2 Computer Programmer Reference Manual.*

NAME

ldsysdump – load multiple floppy sysdump

SYNOPSIS

ldsysdump destination_file

DESCRIPTION

The *ldsysdump* command recombines the multiple floppies taken during a system dump into a single file on the hard disk suitable for use by *crash*(1M). The *destination_file* is the name of the hard disk file into which the floppy data will be loaded.

When invoked, *ldsysdump* begins an interactive procedure that prompts the user to insert the floppies to be loaded. The user has the option of quitting the session at any time. This allows only the portion of the system image needed to be dumped.

EXAMPLES

This example loads the 3 floppies produced via *sysdump* on a machine equipped with 2 MB of memory.

```
$ldsysdump /usr/tmp/cdump

Insert first sysdump floppy.
Enter 'c' to continue, 'q' to quit: c

Loading sysdump
.....
.....

Insert next sysdump floppy.
Enter 'c' to continue, 'q' to quit: c

Loading more sysdump
.....
.....

Insert next sysdump floppy.
Enter 'c' to continue, 'q' to quit: c

Loading more sysdump
.....
.....

3 Sysdump files coalesced, 4096 (512 byte) blocks
$
```

FILES

/dev/dsk/c0d0s6 device used for floppy access

SEE ALSO

crash(1M), sysdump(8).

DIAGNOSTICS

For reversible errors a message will be printed and the user will be allowed to insert a new floppy and continue the session.

WARNINGS

Since the 3B2 computer can be equipped with up to 4 MB of memory, the *destination_file* can become quite large. The file size limit must be set large enough to hold a file of this size.

NAME

led – flash green LED

SYNOPSIS

/etc/led [–f] [–o]

DESCRIPTION

Led is used to turn the green LED (light emitting diode) on. The options are as follows:

–f invokes the *sys3b*(2) system call to set the green LED to a flashing state.

–o invokes *sys3b*(2) system call to set the green LED to a solid on state.

SEE ALSO

sys3b(2) in the *AT&T 3B2 Computer Programmer Reference Manual.*

WARNINGS

This command can only be executed by the super-user.

NAME

link, unlink — exercise link and unlink system calls

SYNOPSIS

/etc/link file1 file2
/etc/unlink file

DESCRIPTION

Link and *unlink* perform their respective system calls on their arguments, abandoning all error checking.

SEE ALSO

rm(1) in the *AT&T 3B2 Computer User Reference Manual.*
link(2), unlink(2) in the *AT&T 3B2 Computer Programmer Reference Manual.*

WARNINGS

These commands may only be executed by the super-user, who (it is hoped) knows what he or she is doing.

NAME

lpadmin – configure the LP spooling system

SYNOPSIS

/usr/lib/lpadmin **-p**printer [options]
/usr/lib/lpadmin **-x**dest
/usr/lib/lpadmin **-d**[dest]

DESCRIPTION

Lpadmin configures LP spooling systems to describe printers, classes and devices. It is used to add and remove destinations, change membership in classes, change devices for printers, change printer interface programs and to change the system default destination. *Lpadmin* may not be used when the LP scheduler, *lpsched*(1M), is running, except where noted below.

Exactly one of the **-p**, **-d** or **-x** options must be present for every legal invocation of *lpadmin*.

-p*printer* names a *printer* to which all of the *options* below refer. If *printer* does not exist then it will be created.

-x*dest* removes destination *dest* from the LP system. If *dest* is a printer and is the only member of a class, then the class will be deleted, too. No other *options* are allowed with **-x**.

-d[*dest*] makes *dest*, an existing destination, the new system default destination. If *dest* is not supplied, then there is no system default destination. This option may be used when *lpsched*(1M) is running. No other *options* are allowed with **-d**.

The following *options* are only useful with **-p** and may appear in any order. For ease of discussion, the printer will be referred to as *P* below.

-c*class* inserts printer *P* into the specified *class*. *Class* will be created if it does not already exist.

-e*printer* copies an existing *printer's* interface program to be the new interface program for *P*.

-h indicates that the device associated with *P* is hardwired. This *option* is assumed when adding a new printer unless the **-l** *option* is supplied.

-i*interface* establishes a new interface program for *P*. *Interface* is the path name of the new program.

-l indicates that the device associated with *P* is a login terminal. The LP scheduler, *lpsched*, disables all login terminals automatically each time it is started. Before re-enabling *P*, its current *device* should be established using *lpadmin*.

-m*model* selects a model interface program for *P*. *Model* is one of the model interface names supplied with the LP Spooling Utilities (see *Models* below).

-r*class* removes printer *P* from the specified *class*. If *P* is the last member of the *class*, then the *class* will be removed.

-v*device* associates a new *device* with printer *P*. *Device* is the pathname of a file that is writable by *lp*. Note that the same *device* can be associated with more than one *printer*. If only the **-p** and **-v** *options* are supplied, then *lpadmin* may be used while the scheduler is running.

Restrictions.

When creating a new printer, the **-v** option and one of the **-e**, **-i** or **-m** options must be supplied. Only one of the **-e**, **-i** or **-m** options may be supplied. The **-h** and **-l** keyletters are mutually exclusive. Printer and class names may be no longer than 14 characters and must consist entirely of the characters **A-Z**, **a-z**, **0-9** and _ (underscore).

Models.

Model printer interface programs are supplied with the LP Spooling Utilities. They are shell procedures which interface between *lpsched* and devices. All models reside in the directory **/usr/spool/lp/model** and may be used as is with *lpadmin* **-m**. Copies of model interface programs may also be modified and then associated with printers using *lpadmin* **-i**. The following describes the *models* and which may be given on the *lp* command line using the **-o** keyletter:

LQP-40

Letter quality printer using XON/XOFF protocol at 9600 baud.

DQP-10

Dot matrix draft quality printer using XON/XOFF protocol at 9600 baud.

EXAMPLES

1. For a DQP-10 printer named cI8, it will use the DQP-10 model interface after the command:

```
/usr/lib/lpadmin -pcI8 -mdqp10
```

2. A LQP-40 printer called pr1 can be added to the lp configuration with the command:

```
/usr/lib/lpadmin -ppr1 -v/dev/contty -mlqp40
```

FILES

/usr/spool/lp/*

SEE ALSO

accept(1M), lpsched(1M).
enable(1), lp(1), lpstat(1) in the *AT&T 3B2 Computer User Reference Manual.*

NAME

lpsched, lpshut, lpmove — start/stop the LP request scheduler and move requests

SYNOPSIS

/usr/lib/lpsched
/usr/lib/lpshut
/usr/lib/lpmove requests dest
/usr/lib/lpmove dest1 dest2

DESCRIPTION

Lpsched schedules requests taken by *lp*(1) for printing on line printers.

Lpshut shuts down the line printer scheduler. All printers that are printing at the time *lpshut* is invoked will stop printing. Requests that were printing at the time a printer was shut down will be reprinted in their entirety after *lpsched* is started again. All LP commands perform their functions even when *lpsched* is not running.

Lpmove moves requests that were queued by *lp*(1) between LP destinations. This command may be used only when *lpsched* is not running.

The first form of the command moves the named *requests* to the LP destination, *dest*. *Requests* are request ids as returned by *lp*(1). The second form moves all requests for destination *dest1* to destination *dest2*. As a side effect, *lp (1)* will reject requests for *dest1*.

Note that *lpmove* never checks the acceptance status (see *accept*(1M)) for the new destination when moving requests.

FILES

/usr/spool/lp/*

SEE ALSO

accept(1M), lpadmin(1M).
enable(1), lp(1), lpstat(1) in the *AT&T 3B2 Computer User Reference Manual.*

NAME

makefsys – create a file system on a diskette

SYNOPSIS

makefsys

DESCRIPTION

This command, which may be under password control, allows the user to create a file system on a diskette.

The user is asked some questions and then the file system is created. Once created, the diskette is self-identifying.

The *makefsys* command may be put under password control by the use of **admpasswd** contained on the *sysadm*(1) manual page.

The identical function is also available under the *sysadm*(1) command:

sysadm makefsys

SEE ALSO

checkfsys(1M), mountfsys(1M).
sysadm(1) in the *AT&T 3B2 Computer User Reference Manual.*

NAME

mkboot – create an object file in proper format for self-config boot

SYNOPSIS

/etc/mkboot [**-m** master] [**-d** directory] [**-k** kernel.o] driver.o ...

DESCRIPTION

Mkboot accepts object files as input and creates the corresponding specially formatted files for use by the self-config boot. Each object file named must have a corresponding *master*(4) file in the **/etc/master.d** directory. The UNIX system kernel object file is always matched with the name *kernel*. The *master* file named for the remaining object files is derived from the object file name itself--any optional path prefix or ".o" suffix is removed, and the lower case result is used as the *master* file name.

Each applicable *master* file is read and the configuration information associated is extracted. For each object file named, a new file is created containing this configuration information. This new file is written to the **/boot** directory and is given the corresponding device name (in capital letters, and without any ".o" suffix) as the corresponding object file.

The options are:

-m *master* specifies the directory containing the master files to be used for each object file. If this flag is omitted, the **/etc/master.d** directory is used.

-d *directory* specifies the directory to be used for storing the each new object file. If this flag is omitted, the **/boot** directory is used.

-k *kernel.o* specifies the name of the object file for the UNIX operating system. The **master** file name used for this object file is always named *kernel*.

EXAMPLE

mkboot -m newmaster adli.o – will read the file named *adli* from the directory **newmaster** for the **adli** device configuration data, take the file **adli.o** from the current directory and create the formatted file **/boot/ADLI** containing the configuration information for the **adli**.

SEE ALSO

master(4) in the *AT&T 3B2 Computer Programmer Reference Manual.*

DIAGNOSTICS

Most messages should be self-explanatory.

Name.o: not processed; cannot open /etc/master.d/name – The file **name.o** was specified on the command line but there was no master file in the master.d directory for *name.o*

Name.o: not processed – An error has aborted processing for the named object file.

BUGS

Mkboot does not clear its internal buffers between modules. The workspace is initially zero-filled but is not reset between objects. Thus, when *mkboot* is used with multiple arguments, the section alignment gaps (of all but the first object) are filled with random data. This should be kept under consideration when comparing driver objects processed by *mkboot*.

NAME

mkfs – construct a file system

SYNOPSIS

/etc/mkfs special blocks[:i-nodes] [gap blocks/cyl]
/etc/mkfs special proto [gap blocks/cyl]

DESCRIPTION

Mkfs constructs a file system by writing on the special file according to the directions found in the remainder of the command line. The command waits 10 seconds before starting to construct the file system. If the second argument is given as a string of digits, *mkfs* builds a file system with a single empty directory on it. The size of the file system is the value of *blocks* interpreted as a decimal number. This is the number of *physical* disk blocks the file system will occupy. The boot program is left uninitialized. If the optional number of i-nodes is not given, the default is the number of *logical* blocks divided by 4.

If the second argument is a file name that can be opened, *mkfs* assumes it to be a prototype file *proto*, and will take its directions from that file. The prototype file contains tokens separated by spaces or new-lines. The first token is the name of a file to be copied onto block zero as the bootstrap program. The second token is a number specifying the size of the created file system in *physical* disk blocks. Typically it will be the number of blocks on the device, perhaps diminished by space for swapping. The next token is the number of i-nodes in the file system. The maximum number of i-nodes configurable is 65500. The next set of tokens comprise the specification for the root file. File specifications consist of tokens giving the mode, the user ID, the group ID, and the initial contents of the file. The syntax of the contents field depends on the mode.

The mode token for a file is a 6-character string. The first character specifies the type of the file. (The characters **–bcd** specify regular, block special, character special and directory files respectively.) The second character of the type is either **u** or – to specify set-user-id mode or not. The third is **g** or – for the set-group-id mode. The rest of the mode is a 3 digit octal number giving the owner, group, and other read, write, execute permissions (see *chmod*(1)).

Two decimal number tokens come after the mode; they specify the user and group IDs of the owner of the file.

If the file is a regular file, the next token is a path name whence the contents and size are copied. If the file is a block or character special file, two decimal number tokens follow which give the major and minor device numbers. If the file is a directory, *mkfs* makes the entries . and .. and then reads a list of names and (recursively) files specifications for the entries in the directory. The scan is terminated with the token $.

A sample prototype specification follows:

```
/stand/diskboot
4872 110
d--777 3 1
usr     d--777 3 1
        sh      ---755 3 1 /bin/sh
        ken     d--755 6 1
                $
        b0      b--644 3 1 0 0
        c0      c--644 3 1 0 0
        $
```

```
$
```

In both command syntaxes, the rotational *gap* and the number of *blocks/cyl* can be specified. The following values are recommended:

Device	Gap Size	Blks/Cyl	
30M Hard Disk	9	90	
10M Hard Disk	9	72	
72M Hard Disk	9	162	(CDC WREN II)
72aM Hard Disk	9	144	(Micropolis)
72bM Hard Disk	9	198	(Priam)
72cM Hard Disk	9	198	(Fujitsu)
Floppy Disk	1	18	

The *default* will be used if the supplied *gap* and *blocks/cyl* are considered illegal values or if a short argument count occurs. The default value is 400 blocks/cyl and gap size 7.

FILES

/etc/vtoc/*

SEE ALSO

chmod(1) in the *AT&T 3B2 Computer User Reference Manual.*
dir(4), fs(4) in the *AT&T 3B2 Computer Programmer Reference Manual.*

BUGS

If a prototype is used, it is not possible to initialize a file larger than 64K bytes, nor is there a way to specify links.

NAME

mknod – build special file

SYNOPSIS

/etc/mknod name **c** | **b** major minor
/etc/mknod name **p**

DESCRIPTION

Mknod makes a directory entry and corresponding i-node for a special file. The first argument is the *name* of the entry. In the first case, the second is **b** if the special file is block-type (disks, tape) or **c** if it is character-type (other devices). The last two arguments are numbers specifying the *major* device type and the *minor* device (e.g., unit, drive, or line number), which may be either decimal or octal.

The assignment of major device numbers is specific to each system. They have to be dug out of the system source file **conf.c**.

Mknod can also be used to create fifo's (a.k.a named pipes) (second case in *SYNOPSIS* above).

SEE ALSO

mknod(2) in the *AT&T 3B2 Computer Programmer Reference Manual.*

NAME

mkunix − create a bootable kernel namelist file, merging kernel and driver symbol tables

SYNOPSIS

mkunix [namelist] [−**o** newlist]

DESCRIPTION

The *mkunix* command will create a bootable kernel namelist file from the UNIX system kernel file and the object files which were loaded by self-config. Typically, *mkunix* would be run following an auto-config boot with a new system configuration. The resulting a.out file can be used as the namelist file for *ps*, *crash*, etc. In addition, this file may be booted directly, bypassing the self-configuration phase of the boot process (see *3B2boot*(8)). This will save on the order of 30 to 60 seconds at boot time.

Namelist (defaults to the path name specified as the **BOOT** program in the **/etc/system** file) is read to obtain the object, data, and symbol table for the basic kernel. This name, if specified, must be the same as that used in **/etc/system** for the boot line; if not, a warning diagnostic is issued since the resulting namelist file will not be accurate.

The argument *-o newlist* (defaults to **a.out**) is the new file--a bootable image of the currently running UNIX system with the composite symbol table.

SEE ALSO

crash(1M), 3B2boot(8).
nm(1), ps(1) in the *AT&T 3B2 Computer User Reference Manual.*

NAME

mount, umount — mount and dismount file system

SYNOPSIS

/etc/mount [special directory [**-r**]]
/etc/umount special

DESCRIPTION

Mount announces to the system that a removable file system is present on the device *special*. The *directory* must exist already; it becomes the name of the root of the newly mounted file system. The mount system call is used to check the validity of the file system.

These commands maintain a table of mounted devices. If invoked with no arguments, *mount* prints the table.

The optional last argument indicates that the file system is to be mounted read-only. Physically write-protected and magnetic tape file systems must be mounted in this way or errors will occur when access times are updated whether or not any explicit write is attempted.

Umount announces to the system that the removable file system previously mounted on device *special* is to be removed.

FILES

/etc/mnttab mount table

SEE ALSO

fuser(1M), mountall(1M), mountfsys(1M), setmnt(1M), umountall(1M).
mount(2), mnttab(4) in the *AT&T 3B2 Computer Programmer Reference Manual.*

DIAGNOSTICS

If the mount system call fails, *mount* prints an appropriate diagnostic. *Mount* issues a warning if the file system to be mounted is currently mounted under another name.

Umount fails if the special file is not mounted or if it is busy. The file system is busy if it contains an open file or some user's working directory.

BUGS

Some degree of validation is done on the file system; however, it is generally unwise to mount garbage file systems.

NAME

mountall — mount all file systems according to a table

SYNOPSIS

/etc/mountall *file-system-table* ...

DESCRIPTION

This command is executed by the super-user, "root", to mount file systems according to a *file-system-table*. The special file name "-" reads from the standard input.

Before each file system is mounted, it is checked using *fsstat*(1M) to see if it appears mountable. If the file system does not appear to be mountable, it is checked, using *fsck*(1M), before the mount is attempted.

EXAMPLES

The following examples are equivalent:

```
/etc/mountall /etc/fstab /etc/fstab2

/etc/mountall - /etc/fstab2 < /etc/fstab
```

FILES

File-system-table format:

column 1	block special file name of file system
column 2	mount-point directory
column 3	"-r" if to be mounted readonly
column 4+	ignored

White-space separates columns. Lines beginning with "#" are comments. Empty lines are ignored.

A typical file-system-table might read:

```
/dev/dsk/c1d0s2    /usr
```

SEE ALSO

fsck(1M), fsstat(1M), mount(1M).
sysadm(1) in the *AT&T 3B2 Computer User Reference Manual.*

DIAGNOSTICS

No messages are printed if the file systems are mountable and clean.

Error and warning messages come from *fsck*(1M), *fsstat*(1M), and *mount*(1M).

NAME

mountfsys, umountfsys — mount (unmount) a diskette file system

SYNOPSIS

mountfsys [−y] [−r]
umountfsys [−y]

DESCRIPTION

The *mountfsys* command mounts a file system that is on a removable disk so that the user can read and write on it. The options provide the following:

−r the file system is mounted read-only.

−y suppresses any questions asked the user during mounting or unmounting.

The *umountfsys* command unmounts the file system.

By default, the user is told the name of the file system on the disk and asked if the file system should be mounted. The optional −y argument supresses the questions and mounts or unmounts the file system immediately.

The commands *mountfsys* and *unmountfsys* can be password controlled. See *sysadm*(1), **admpasswd** sub-command.

SEE ALSO

checkfsys(1M), makefsys(1M).
sysadm(1) in the *AT&T 3B2 Computer User Reference Manual.*

WARNING

ONCE THE DISK IS MOUNTED IT MUST NOT BE REMOVED FROM THE DISK DRIVE UNTIL IT HAS BEEN UNMOUNTED!

Removing the disk while it is still mounted can cause the data on the disk to be damaged beyond repair.

The *checkfsys*(1M) command can be used to check for and optionally repair a damaged file system on a removable disk.

The identical functions are also available under the *sysadm* commands:

sysadm mountfsys
sysadm umountfsys

BUGS

The *mountfsys* command should refuse to mount a file system that has been removed without unmounting or which was mounted at the time of a system crash.

The hardware should prevent removal of a disk whenever it is opened or mounted.

It should be possible to detect that a write-protected disk and only mount it as "read-only".

A file system that has no label cannot be mounted.

NAME

mvdir – move a directory

SYNOPSIS

/etc/mvdir dirname name

DESCRIPTION

Mvdir moves directories within a file system. *Dirname* must be a directory; *name* must not exist. Neither name may be a sub-set of the other (/x/y cannot be moved to /x/y/z, nor vice versa).

SEE ALSO

mkdir(1), cp(1) in the *AT&T 3B2 Computer User Reference Manual.*

WARNINGS

Only super-user can use *mvdir*.

NAME

ncheck – generate names from i-numbers

SYNOPSIS

/etc/ncheck [–i numbers] [–a] [–s] [file-system]

DESCRIPTION

Ncheck with no argument generates a path-name vs. i-number list of all files on a set of default file systems. Names of directory files are followed by **/.**.

The options are as follows:

–i reduces the report to only those files whose i-numbers follow.

–a allows printing of the names . and .., which are ordinarily suppressed.

–s reduces the report to special files and files with set-user-ID mode; it is intended to discover concealed violations of security policy.

A file system may be specified.

The report should be sorted so that it is more useful.

SEE ALSO

fsck(1M).
sort(1) in the *AT&T 3B2 Computer User Reference Manual.*

DIAGNOSTICS

When the file system structure is improper, ?? denotes the "parent" of a parentless file and a path-name beginning with ... denotes a loop.

NAME

newboot − load lboot and mboot onto the device boot partition

SYNOPSIS

/etc/newboot [**−y**] /lib/lboot /lib/mboot boot-special
/etc/newboot [**−y**] /lib/olboot /lib/mboot boot-special

DESCRIPTION

Newboot replaces the *lboot* and *mboot* files on the given *boot-special* section of a hard disk. In the case of a floppy disk, *newboot* places *olboot* and *mboot* files on the *boot-special* section of a disk. A confirmation is required before the *boot-special* file is overwritten if **−y** is not specified.

Mboot is the 512-byte micro boot file loaded by the boot device firmware and executed to load the larger *lboot* file.

Lboot is a file containing the boot program that is loaded by *mboot* and executed to boot the UNIX system.

Olboot is a file containing the boot program on a floppy diskette that is loaded by *mboot* and executed to boot the absolute operating system.

SEE ALSO

dd(1M), mkboot(1M).

DIAGNOSTICS

Can't open file FILE FILE not found.

WARNINGS

Installing a bad *lboot* or *mboot* may make the affected disk pack unbootable. Be sure you have a good backup disk before *newboot* is run.

NAME

newgrp – log in to a new group

SYNOPSIS

newgrp [–] [group]

DESCRIPTION

Newgrp changes a user's group identification. The user remains logged in and the current directory is unchanged, but calculations of access permissions to files are performed with respect to the new real and effective group IDs. The user is always given a new shell, replacing the current shell, by *newgrp*, regardless of whether it terminated successfully or due to an error condition (i.e., unknown group).

Exported variables retain their values after invoking *newgrp*; however, all unexported variables are either reset to their default value or set to null. System variables (such as PS1, PS2, PATH, MAIL, and HOME), unless exported by the system or explicitly exported by the user, are reset to default values. For example, a user has a primary prompt string (**PS1**) other than **$** (default) and has not exported **PS1**. After an invocation of *newgrp* , successful or not, their **PS1** will now be set to the default prompt string **$**. Note that the shell command *export* (see *sh*(1)) is the method to export variables so that they retain their assigned value when invoking new shells.

With no arguments, *newgrp* changes the group identification back to the group specified in the user's password file entry.

If the first argument to *newgrp* is a –, the environment is changed to what would be expected if the user actually logged in again.

A password is demanded if the group has a password and the user does not, or if the group has a password and the user is not listed in **/etc/group** as being a member of that group.

FILES

/etc/group	system's group file
/etc/passwd	system's password file

SEE ALSO

environ(5), group(4), passwd(4), in the *AT&T 3B2 Computer Programmer Reference Manual.*
login(1), sh(1) in the *AT&T 3B2 Computer User Reference Manual.*

BUGS

There is no convenient way to enter a password into **/etc/group**. Use of group passwords is not encouraged, because, by their very nature, they encourage poor security practices. Group passwords may disappear in the future.

NAME

powerdown — stop all processes and turn off the power

SYNOPSIS

powerdown [−y | −Y]

DESCRIPTION

This command brings the system to a state where nothing is running and then turns off the power.

By default, the user is asked questions that control how much warning the other users are given. The options:

−y prevents the questions from being asked and just gives the warning messages. There is a 60 second pause between the warning messages.

−Y is the same as −y except it has no pause between messages. It is the fastest way to bring the system down.

Password control can be instituted on this command. See *sysadm*(1), **admpasswd** sub-command.

The identical function is also available under the *sysadm* command:

sysadm powerdown

EXAMPLES

some-long-running-command; powerdown −y

The first command is run to completion and then the machine turns off. This is useful for, say, formatting a document to the printer at the end of a day.

FILES

/etc/shutdown - does the work

SEE ALSO

shutdown(1M).

sysadm(1) in the *AT&T 3B2 Computer User Reference Manual.*

NAME

prfld, prfstat, prfdc, prfsnap, prfpr — operating system profiler

SYNOPSIS

/etc/prfld [namelist]
/etc/prfstat on
/etc/prfstat off
/etc/prfdc file [period [off_hour]]
/etc/prfsnap file
/etc/prfpr file [cutoff [namelist]]

DESCRIPTION

Prfld, *prfstat*, *prfdc*, *prfsnap*, and *prfpr* form a system of programs to facilitate an activity study of the UNIX operating system.

Prfld is used to initialize the recording mechanism in the system. It generates a table containing the starting address of each system subroutine as extracted from *namelist*.

Prfstat is used to enable or disable the sampling mechanism. Profiler overhead is less than 1% as calculated for 500 text addresses. *Prfstat* will also reveal the number of text addresses being measured.

Prfdc and *prfsnap* perform the data collection function of the profiler by copying the current value of all the text address counters to a file where the data can be analyzed. *Prfdc* will store the counters into *file* every *period* minutes and will turn off at *off_hour* (valid values for *off_hour* are **0**—**24**). *Prfsnap* collects data at the time of invocation only, appending the counter values to *file*.

Prfpr formats the data collected by *prfdc* or *prfsnap*. Each text address is converted to the nearest text symbol (as found in *namelist*) and is printed if the percent activity for that range is greater than *cutoff*.

FILES

/dev/prf	interface to profile data and text addresses
/unix	default for namelist file

NAME

/etc/prtconf - print system configuration

SYNOPSIS

/etc/prtconf

DESCRIPTION

The *prtconf* command prints the system configuration information when executed. The configuration information is displayed every time the system is initialized to multi-user mode. The information printed includes the memory and peripheral configuration.

EXAMPLES

To print the configuration of the 3B2 Computer:

```
/etc/prtconf

AT&T 3B2 SYSTEM CONFIGURATION:

Memory size: 1024 kilobytes
System Peripherals:

SBD         /*System Board*/
FD5         /*Floppy Disk Drive*/
HD30        /*30 Megabyte Hard Disk Drive*/
PORTS       /*Expansion Ports Feature Card*/
CTC         /*Cartridge Tape Controller Card*/
```

FILES

/etc/rc.d/sysetup

SEE ALSO

3B2 Computer System Administration Reference Manual.

BUGS

The output in response to the *prtconf* command is not aligned properly.

NAME

prtvtoc – print the volume table of contents of a block device

SYNOPSIS

prtvtoc **–h** –s [–tfstab] /dev/rdsk/c?d?s?

DESCRIPTION

Prtvtoc allows the contents of the VTOC (volume table of contents) to be viewed by the user logged in as root for reference or verification.

The *device* name must be a raw device in the form of */dev/rdsk/c?d?s6*.

The following options apply to *prtvtoc*:

–h prints the VTOC information without headers.

–s prints the VTOC information with column headers only.

–t the *fstab* file specified will be used in place of /etc/fstab in determining which directories partitions are mounted as.

NAME

pump – Download B16 or X86 a.out file to a peripheral board

SYNOPSIS

pump /dev/devname file

DESCRIPTION

The *pump* command will read a B16 or X86 **a.out** file's sections into a buffer according to the physical address of the section. *Pump* expects a section in the **a.out** file called ".start". Once it has found this section, *pump* will inform the peripheral to start executing at the address that it found in ".start" after it has downloaded the **a.out** file.

Error Messages

Pump error: UNIX error number – Can't get status of /dev/devname

There may be no **/dev/devname.**

Pump Error: error number – ioctl call

The ioctl call failed. The error number returned can be a UNIX system error number or, in the case of the NI, an error number of 208. Error number 208 is a timeout message. The peripheral board did not respond in time to the request made of it [this is not the only error, see *intro*(2) for a complete list].

Can't open a.out filename for reading!

The error may be that there is no such file or the permissions are such that the file cannot be read [see *chmod*(1)].

Error: Object file is not in b16 or x86 common object format

The file to be downloaded to the peripheral is not a B16 or X86 **a.out** file.

Section size is too big for the buffer

The **a.out** file may be greater that the 32K bytes that is the limit of RAM on the peripheral.

Error: No section name called .start

Pump needs ".start" for the starting address that the peripheral needs to execute the downloaded code.

Pump: /dev/devname returned a CIO FAULT during phase

The peripheral encountered a hardware fault during one of the phases of the pump. Phase is one of the following:

reset	This phase will cause a hardware reset on the peripheral.
downlaod	This phase will download the **a.out** file to the peripheral.

force call to function
: This phase will inform the peripheral to start executing at the address found in the ".start" section.

sysgen
: This phase will *sysgen* the peripheral. It allows normal functioning of the peripheral to occur.

Pump: /dev/devname returned a CIO Invalid Queue Entry during phase

The peripheral did not understand the command phase that was issued by *pump*.

Pump: /dev/devname did not respond during phase

The UNIX system driver called may not have understood the command.

Pump: A timeout has occurred on /dev/devname during phase

The peripheral did not respond to one of the commands given.

Pump: There was no return code for /dev/devname during phase

The return code that was given may have been corrupted.

SEE ALSO

intro(2), a.out(4) in the *AT&T 3B2 Computer Programmer Reference Manual.*

NAME

pwck, grpck — password/group file checkers

SYNOPSIS

/etc/pwck [file]
/etc/grpck [file]

DESCRIPTION

Pwck scans the password file and notes any inconsistencies. The checks include validation of the number of fields, login name, user ID, group ID, and whether the login directory and optional program name exist. The default password file is **/etc/passwd**.

Grpck verifies all entries in the group file. This verification includes a check of the number of fields, group name, group ID, and whether all login names appear in the password file. The default group file is **/etc/group**.

FILES

/etc/group
/etc/passwd

SEE ALSO

group(4), passwd(4) in the *AT&T 3B2 Computer Programmer Reference Manual.*

DIAGNOSTICS

Group entries in **/etc/group** with no login names are flagged.

NAME

rc0 – run commands performed to stop the operating system

SYNOPSIS

/etc/rc0

DESCRIPTION

This file is executed at each system state change that needs to have the system in an inactive state. It is responsible for those actions that bring the system to a quiescent state, traditionally called "shut down".

Normally, init state 0 is taken to mean "stop the operating system", so the inittab entry might read:

r0:0:wait:/etc/rc0 > /dev/console 2>&1

The recommended sequence for */etc/rc0* is:

Stop System Services, Demons, Accounting, etc.

The */etc/rc0* command executes the files found in the directory **/etc/shutdown.d**. Each of these files terminates some system service. When new services are added that should be terminated when the system is shut down, the appropriate files are installed in **/etc/shutdown.d**.

Terminate Processes

SIGTERM signals are sent to all running processes by *killall*(1M). Processes stop themselves cleanly if sent SIGTERM.

Kill Processes

SIGKILL signals are sent to all remaining processes; no process can resist SIGKILL.

At this point the only processes left are those associated with */etc/rc0* and processes 0 and 1, which are special to the operating system.

Unmount All File Systems

Only the root file system (/) remains mounted.

If the **/etc/inittab** has not defined any other actions to be performed for init stat 0, then the operating system should have nothing to do. It should not be possible to get the system's attention. The only thing that can be done is to turn off the power or possibly get the attention of a firmware monitor.

EXAMPLES

On machines where it is possible to turn off the power under program control, or to reboot, or go to a firmware monitor, one could set up the **/etc/inittab** like this:

```
fl:056:wait:/etc/flash-power-light
r0:056:wait:/etc/rc0 >/dev/console 2>&1
po:0:wait:/etc/poweroff
fw:5:wait:/etc/firmware
rb:6:wait:/etc/reboot
```

The following are prototypical files found in **/etc/shutdown.d**.

errdemon

```
#   Terminate the error logging process
/etc/errstop
```

procacct

```
#   Terminate process accounting
/usr/lib/acct/shutacct
```

FILES

The advice in *letc/rc2*(1M) is appropriate for files in /etc/**shutdown.d**.

Files in /etc/**shutdown.d** that begin with a dot, ".", will not be executed. This feature can be used to "hide" files that are not to be executed for the time being without removing them.

SEE ALSO

killall(1M), rc2(1M), shutdown(1M).

NAME

rc2 − run commands performed for multi-user environment

SYNOPSIS

/etc/rc2

DESCRIPTION

This file is executed at each system state change that goes to one of the numbered states (0 through 6) and is responsible for those initializations that bring the system to a ready-to-use state, traditionally called "multi-user".

The actions performed by */etc/rc2* are found in files in the directory **/etc/rc.d**. These files are executed by **/bin/sh** in ascii sort sequence order. Thus that names of the files are significant if there are interdependencies between them. When functions are added that need to be initialized when the system goes multi-user, an appropriate file should be added in **/etc/rc.d**.

EXAMPLES

The following are prototypical files found in **/etc/rc.d**.

MOUNTFILESYS

```
#   Set up and mount file systems

cd /
> /etc/mnttab
/etc/devnm /  | grep −v swap | grep  −v root | /etc/setmnt

# clean up /tmp
rm −rf /tmp
mkdir /tmp

mount /dev/usr /usr
mount /dev/fs1 /fs1
```

uucp

```
#   clean-up uucp locks, status, and temporary files


rm −f /usr/spool/uucp/LCK* /usr/spool/uucp/STST.* /usr/spool/uucp/TM.
```

The file **etc/TIMEZONE** is included early in */etc/rc2*, thus establishing the default time zone for all commands that follow.

Some hints about files in **/etc/rc.d**: If your world is simple, you can probably get away with a simple naming rule such as "all important things are named with capital letters, everything else starts with lower case." If you have lots of interdependencies and orders are important, using the first character as a sequence indicator may help. Thus files starting with the following characters would be:

[0-9]. very early
[A-Z]. early
[a-n]. later
[o-z]. last

So that the files in **/etc/rc.d** might be named:

3.mountfilesys
B.errdemon
c.uucp

r.cron

Files in **/etc/rc.d** that begin with a dot, ".", will not be executed. This feature can be used to "hide" files that are not to be executed for time being without removing them.

SEE ALSO

/etc/shutdown(1M).

NAME

sadp − disk access profiler

SYNOPSIS

sadp [**−th**] [**−d** device[−drive]] s [n]

DESCRIPTION

Sadp reports disk access location and seek distance, in tabular or histogram form. It samples disk activity once every second during an interval of *s* seconds. This is done repeatedly if *n* is specified. Cylinder usage and disk distance are recorded in units of 8 cylinders.

Valid values of *device* are **hdsk** for integral disk and **fdsk** for integral floppy. *Drive* specifies the disk drives and it may be:

a drive number in the range supported by *device*,
two numbers separated by a minus (indicating an inclusive range),

or

a list of drive numbers separated by commas.

Up to 8 disk drives may be reported. The **−d** option may be omitted, if only one *device* is present.

The **−t** flag causes the data to be reported in tabular form. The **−h** flag produces a histogram on the printer of the data. Default is **−t**.

EXAMPLE

The command:

sadp −d rp06−0 900 4

will generate 4 tabular reports, each describing cylinder usage and seek distance of rp06 disk drive 0 during a 15-minute interval.

FILES

/dev/kmem

SEE ALSO

mem(7).

NAME

sa1, sa2, sadc – system activity report package

SYNOPSIS

/usr/lib/sa/sadc [t n] [ofile]

/usr/lib/sa/sa1 [t n]

/usr/lib/sa/sa2 [–**ubdycwaqvmA**] [–s time] [–e time] [–i sec]

DESCRIPTION

System activity data can be accessed at the special request of a user (see *sar*(1)) and automatically on a routine basis as described here. The operating system contains a number of counters that are incremented as various system actions occur. These include CPU utilization counters, buffer usage counters, disk and tape I/O activity counters, TTY device activity counters, switching and system-call counters, file-access counters, queue activity counters, and counters for inter-process communications.

Sadc and shell procedures, *sa1* and *sa2*, are used to sample, save, and process this data.

Sadc, the data collector, samples system data *n* times every *t* seconds and writes in binary format to *ofile* or to standard output. If *t* and *n* are omitted, a special record is written. This facility is used at system boot time to mark the time at which the counters restart from zero. The **/etc/rc** entry:

```
su sys –c "/usr/lib/sa/sadc /usr/adm/sa/sa`date +%d`"
```

writes the special record to the daily data file to mark the system restart.

The shell script *sa1*, a variant of *sadc*, is used to collect and store data in binary file **/usr/adm/sa/sa***dd* where *dd* is the current day. The arguments *t* and *n* cause records to be written *n* times at an interval of *t* seconds, or once if omitted. The entries in **/usr/spool/cron/crontabs/sys** (see *cron*(1M)):

```
0 * * * 0,6 /usr/lib/sa/sa1
0 8–17 * * 1–5 /usr/lib/sa/sa1 1200 3
0 18–7 * * 1–5 /usr/lib/sa/sa1
```

will produce records every 20 minutes during working hours and hourly otherwise.

The shell script *sa2*, a variant of *sar*(1), writes a daily report in file **/usr/adm/sa/sar***dd*. The options are explained in *sar*(1). The **/usr/spool/cron/crontabs/sys** entry:

```
5 18 * * 1–5 /usr/lib/sa/sa2 –s 8:00 –e 18:01 –i 3600 –A
```

will report important activities hourly during the working day.

The structure of the binary daily data file is:

```
struct sa {
        struct sysinfo si;  /* see /usr/include/sys/sysinfo.h */
        int  szi-node;      /* current entries of i-node table */
        int  szfile;        /* current entries of file table */
        int  sztext;        /* current entries of text table */
        int  szproc;        /* current entries of proc table */
        int  mszi-node;     /* size of i-node table */
        int  mszfile;       /* size of file table */
        int  msztext;       /* size of text table */
        int  mszproc;       /* size of proc table */
        long  i-nodeovf;    /* cumul. overflows of i-node table */
        long  fileovf;      /* cumul. overflows of file table */
        long  textovf;      /* cumul. overflows of text table */
        long  procovf;      /* cumul. overflows of proc table */
        time_t  ts;         /* time stamp, seconds */
        long  devio[NDEVS][4];   /* device info for up to NDEVS units */
#define IO_OPS          0       /* cumul. I/O requests */
#define IO_BCNT         1       /* cumul. blocks transferred */
#define IO_ACT          2       /* cumul. drive busy time in ticks */
#define IO_RESP         3       /* cumul. I/O resp time in ticks */
};
```

FILES

/usr/adm/sa/sa*dd*	daily data file
/usr/adm/sa/sar*dd*	daily report file
/tmp/sa.adrfl	address file

SEE ALSO

sag(1G), sar(1), timex(1).
cron(1M) in the *AT&T 3B2 Computer User Reference Manual.*

NAME

setclk – set clock

SYNOPSIS

setclk

DESCRIPTION

The *setclk* command checks the NVRAM only for the correct date. If the date is wrong *setclk* promplts the user to use *sysadm* **datetime** [see *sysadm*(1)] for the proper setting of the hardware clock. *Setclk* is executed through the inittab of the system initialization time.

SEE ALSO

sysadm(1) in the *AT&T 3B2 Computer User Reference Manual.*

NAME

setmnt – establish mount table

SYNOPSIS

/etc/setmnt

DESCRIPTION

Setmnt creates the **/etc/mnttab** table which is needed for both the *mount*(1M) and *umount* commands. *Setmnt* reads standard input and creates a *mnttab* entry for each line. Input lines have the format:

filesys node

where *filesys* is the name of the file system's *special file* (e.g., c?d?s?) and *node* (mount point) is the root name of that file system. Thus *filesys* and *node* become the first two strings in the mount table entry.

EXAMPLE

```
#setmnt<CR>
c1d0s0 /<CR>
c1d0s2 /usr<CR>
<CTRL d>
```

FILES

/etc/mnttab

SEE ALSO

mount(1M).

BUGS

Problems may occur if *filesys* or *node* are longer than 32 characters.
Setmnt silently enforces an upper limit on the maximum number of *mnttab* entries.

NAME

shutdown – shut down system

SYNOPSIS

/etc/shutdown [**–y**] [**–g***grace-period* [**–i***init-state*]

DESCRIPTION

This command is executed by the super-user, "root", to change the state of the machine. By default, it brings the system to a state where only the console has access to the UNIX system. This "root-only" state is traditionally called "single-user".

The command sends a warning message and a final message before it starts actual shutdown activities. By default, the command asks for confirmation before it starts shutting down daemons and killing processes. The options are used as follows:

–y pre-answers the question so the command can be run without user intervention. By default, the time periods between the warning message and the final message and between the final message and the confirmation are 60 seconds.

–g*grace-period*
allows the super-user to specify a different number of seconds.

–i*init-state*
specifies the state that *init*(1M) is to be put in following the warnings, if any. By default, init state "s" is used.

NOTE: THIS VERSION OF SHUTDOWN IS DIFFERENT FROM PREVIOUS VERSIONS.

In the past, the *shutdown* procedure performed process killing and file system unmounts before changing init state. This proved unreliable. Now, the new init state defines what state the machine is to be in and is responsible for making it that way. Recommended definitions are:

state 0
Shut the machine down so it is safe to remove the power. Have the machine remove power if it can.

state 1 or state s
Single user [1 is translated to s in **/etc/inittab**].

state 2
Bring machine to state traditionally called multi-user.

state 3 and state 4
These are user defined. They are also used to fix bad blocks that produce errors while in single user mode.

state 5
Stop the UNIX system and go to the firmware monitor.

state 6
Stop the UNIX system and reboot.

SEE ALSO

init(1M), rc0(1M), rc2(1M).

NAME

su – become super-user or another user

SYNOPSIS

su [–] [name [arg ...]]

DESCRIPTION

Su allows one to become another user without logging off. The default user *name* is **root** (i.e., super-user).

To use *su*, the appropriate password must be supplied (unless one is already **root**). If the password is correct, *su* will execute a new shell with the real and effective user ID set to that of the specified user. The new shell will be the optional program named in the shell field of the specified user's password file entry (see *passwd*(4)), or **/bin/sh** if none is specified (see *sh*(1)). To restore normal user ID privileges, type an **EOF** (*cntrl-d*) to the new shell.

Any additional arguments given on the command line are passed to the program invoked as the shell. When using programs like *sh*(1), an *arg* of the form –c *string* executes *string* via the shell and an arg of –r will give the user a restricted shell.

The following statements are true only if the optional program named in the shell field of the specified user's password file entry is like *sh*(1). If the first argument to *su* is a –, the environment will be changed to what would be expected if the user actually logged in as the specified user. This is done by invoking the program used as the shell with an *arg0* value whose first character is –, thus causing first the system's profile (**/etc/profile**) and then the specified user's profile (**.profile** in the new HOME directory) to be executed. Otherwise, the environment is passed along with the possible exception of **$PATH**, which is set to **/bin:/etc:/usr/bin** for **root**. Note that if the optional program used as the shell is **/bin/sh**, the user's **.profile** can check *arg0* for **–sh** or **–su** to determine if it was invoked by *login* or *su*, respectively. If the user's program is other than **/bin/sh**, then **.profile** is invoked with an *arg0* of -*program* by *login*(1).

All attempts to become another user using *su* are logged in the log file **/usr/adm/sulog**.

EXAMPLES

To become user **bin** while retaining your previously exported environment, execute:

```
su bin
```

To become user **bin** but change the environment to what would be expected if **bin** had originally logged in, execute:

```
su - bin
```

To execute *command* with the temporary environment and permissions of user **bin**, type:

```
su - bin -c "command args"
```

FILES

/etc/passwd	system's password file
/etc/profile	system's profile
$HOME/.profile	user's profile
/usr/adm/sulog	log file

SEE ALSO

env(1), login(1), sh(1) in the *AT&T 3B2 Computer User Reference Manual.*
environ(5), passwd(4), profile(4) in the *AT&T 3B2 Computer Programmer Reference Manual.*

NAME

sync – update the super block

SYNOPSIS

sync

DESCRIPTION

Sync executes the *sync* system primitive. If the system is to be stopped, *sync* must be called to insure file system integrity. It will flush all previously unwritten system buffers out to disk, thus assuring that all file modifications up to that point will be saved. See *sync*(2) for details.

SEE ALSO

sync(2) in the *AT&T 3B2 Computer Programmer Reference Manual.*

NAME

sysdef – system definition

SYNOPSIS

/etc/sysdef [opsys [master.d]]

DESCRIPTION

Sysdef analyzes the named operating system file (opsys) and extracts configuration information. The operating system file must be an "absolute" boot file [see *mkunix*(1M)]. All hardware devices, their local bus addresses, and unit count, as well as pseudo devices, system devices, and loadable modules are listed. In addition, the values of all tunable parameters are listed. The output of *sysdef* is in tabular form.

FILES

/unix	default operating system file
/etc/master.d/*	default directory containing master files

SEE ALSO

mkunix(1M).

master(4), nlist(3C) in the *AT&T 3B2 Computer Programmer Reference Manual.*

DIAGNOSTICS

internal name list overflow

if the master table contains more than an internally specified number of entries for use by *nlist*(3C).

NAME

tic – terminfo compiler

SYNOPSIS

tic [**-v**[*n*]] file ...

DESCRIPTION

Tic translates terminfo files from the source format into the compiled format. The results are placed in the directory **/usr/lib/terminfo**.

The **-v** (verbose) option causes *tic* to output trace information showing its progress. If the optional integer is appended, the level of verbosity can be increased.

Tic compiles all terminfo descriptions in the given files. When a **use=** field is discovered, *tic* searches first the current file, then the master file, which is "./terminfo.src".

If the environment variable TERMINFO is set, the results are placed there instead of **/usr/lib/terminfo**.

Some limitations: total compiled entries cannot exceed 4096 bytes. The name field cannot exceed 128 bytes.

FILES

/usr/lib/terminfo/*/* compiled terminal capability data base

SEE ALSO

curses(3X), terminfo(4).

BUGS

Instead of searching **./terminfo.src**, it should check for an existing compiled entry.

NAME

uadmin – administrative control

SYNOPSIS

uadmin cmd fcn

DESCRIPTION

The *uadmin* command provides control for basic administrative functions. This command is tightly coupled to the system administration procedures and is not intended for general use. It may be invoked only by the super-user.

The arguments are converted to integers and passed to the *uadmin* system call.

SEE ALSO

uadmin(2) in the *AT&T 3B2 Computer Programmer Reference Manual.*

NAME

umountall – unmount file systems

SYNOPSIS

umountall [-k]

DESCRIPTION

The *umountall* command is executed by the super-user, "root", to unmount all currently mounted file systems except the root file system.

The **-k** option causes *fuser*(1M) to send a SIGKILL signal to all processes that have files open in each file system before it is unmounted. Without **-k** it is possible that an unmount may fail because the file system is busy.

EXAMPLES

/etc/umountall

/etc/umountall -k

SEE ALSO

fuser(1M).
signal(2) in the *AT&T 3B2 Computer Programmer Reference Manual.*

NAME

uucheck – check the uucp directories and permissions file

SYNOPSIS

/usr/lib/uucp/uucheck [**-v**] [**-x** debug_level]

DESCRIPTION

Uucheck checks for the presence of the *uucp* system required files and directories. Within the *uucp* makefile, it is executed before the installation takes place. It also checks for some obvious errors in the Permissions file (**/usr/lib/uucp/Permissions**). When executed with the **-v** option, it gives a detailed explanation of how the uucp programs will interpret the Permissions file. The **-x** option is used for debugging; it takes a single digit, higher numbers for more detail. Note, however, that compiling *uucp* with the **-DSMALL** option will result in little debugging output.

Note that *uucheck* can only be used by the super-user or *uucp*.

FILES

/usr/lib/uucp/Systems
/usr/lib/uucp/Permissions
/usr/lib/uucp/Devices
/usr/lib/uucp/Maxuuscheds
/usr/lib/uucp/Maxuuxqts
/usr/spool/uucp/*
/usr/spool/locks/LCK*
/usr/spool/uucppublic/*

SEE ALSO

uucico(1M), uusched(1M).
uucp(1C), uustat(1C), uux(1C) in the *AT&T 3B2 Computer User reference Manual.*

BUGS

The program does not check file/directory modes or some errors in the Permissions file such as duplicate login or machine name.

NAME

uucico − file transport program for the uucp system

SYNOPSIS

/usr/lib/uucp/uucico [**−r** role_number] [**−x** debug_level]
[**−d** spool_directory] **−s** system_name

DESCRIPTION

Uucico is the file transport program for *uucp* work file transfers. Role numbers for the **−r** are the digit 1 for master mode or 0 for slave mode (default). The **−r** option should be specified as the digit 1 for master mode when *uucico* is started by a program or *cron*. *Uux* and *uucp* both queue jobs that will be transferred by *uucico*. It is normally started by the scheduler, *uusched* but can be started manually; this is done for debugging. For example, the shell *Uutry* starts *uucico* with debugging turned on. A single digit must be used for the **−x** option with higher numbers for more debugging.

FILES

/usr/lib/uucp/Systems
/usr/lib/uucp/Permissions
/usr/lib/uucp/Devices
/usr/lib/uucp/Maxuuxqts
/usr/lib/uucp/Maxuuscheds
/usr/spool/uucp/*
/usr/spool/locks/LCK*
/usr/spool/uucppublic/*

SEE ALSO

cron(1M), uusched(1M), Uutry(1M).
uucp(1C), uustat(1C), uux(1C) in the *AT&T 3B2 Computer User Reference Manual*.

NAME

uucleanup − uucp spool directory clean-up

SYNOPSIS

/usr/lib/uucp/uucleanup [−C*time*] [−W*time*] [−X*time*] [−m*string*] [−o*time*] [−s*system*]

DESCRIPTION

Uucleanup will scan the spool directories for old files and take appropriate action to remove them in a useful way: Inform the requestor of send/receive requests for systems that can not be reached. Return mail, which cannot be delivered, to the sender. Delete or execute rnews for rnews type files (depending on where the news originated−−locally or remotely). Remove all other files. In addition, there is provision to warn users of requests that have been waiting for a given number of days (default 1). Note that *uucleanup* will process as if all option *times* were specified to the default values unless *time* is specifically set.

The following options are available.

−C*time* Any **C.** files greater or equal to *time* days old will be removed with appropriate information to the requestor. (default 7 days)

−D*time* Any **D.** files greater or equal to *time* days old will be removed. An attempt will be made to deliver mail messages and execute rnews when appropriate. (default 7 days)

−W*time* Any **C.** files equal to *time* days old will cause a mail message to be sent to the requestor warning about the delay in contacting the remote. The message includes the *JOBID*, and in the case of mail, the mail message. The administrator may include a message line telling whom to call to check the problem (−m option). (default 1 day)

−X*time* Any **X.** files greater or equal to *time* days old will be removed. The **D.** files are probably not present (if they were, the **X.** could get executed). But if there are **D.** files, they will be taken care of by D. processing. (default 2 days)

−m*string* This line will be included in the warning message generated by the −**W** option.

−o*time* Other files whose age is more than *time* days will be deleted. (default 2 days) The default line is "See your local administrator to locate the problem".

−s*system* Execute for *system* spool directory only.

−x*debug_level*
The −x debug level is a single digit between 0 and 9; higher numbers give more detailed debugging information. Note, however, that if *uucp* is compiled with the −**DSMALL** option, there will be very little debugging output.

This program is typically started by the shell *uudemon.cleanup*, which should be started by *cron*(1M).

FILES

/usr/lib/uucp	directory with commands used by *uucleanup* internally
/usr/spool/uucp	spool directory

SEE ALSO

cron(1M).

uucp(1C), uux(1C) in the *AT&T 3B2 Computer User Reference Manual.*

NAME

uugetty – set terminal type, modes, speed, and line discipline

SYNOPSIS

/usr/lib/uucp/getty [**-h**] [**-t** timeout] [**-r**] line
[speed [type [linedisc]]]
/usr/lib/uucp/getty **-c** file

DESCRIPTION

Uugetty is identical to *getty*(1M) but changes have been made to support using the line for *uucico*, *cu*, and *ct*; that is, the line can be used in both directions. The *uugetty* will allow users to login, but if the line is free, *uucico*, *cu*, or *ct* can use it for dialing out. The implementation depends on the fact that *uucico*, *cu*, and *ct* create lock files when devices are used. When the "open()" returns (or the first character is read when **-r** option is used), the status of the lock file indicates whether the line is being used by *uucico*, *cu*, *ct*, or someone trying to login. Note that in the **-r** case, several <carriage-return> characters may be required before the login message is output. The human users will be able to handle this slight inconvenience. *Uucico* trying to login will have to be told by using the following login script:

"" \r\d\r\d\r\d\r in:--in: . . .

where the . . . is whatever would normally be used for the login sequence.

Here is an **/etc/inittab** entry using *uugetty* on an 801/212 dialer:

30:2:respawn:/usr/lib/uucp/uugetty -t 60 cul04 1200

The line name, cul04, is the name that appears in the **/usr/lib/uucp/Devices** file for the 212 dialer.

An entry for an intelligent modem or direct line that has a *uugetty* on each end must use the **-r** option. (This causes *uugetty* to wait to read a character before it puts out the login message, thus preventing two uugettys from looping.) If there is a *uugetty* on one end of a direct line, there must be a *uugetty* on the other end as well. Here is an **/etc/inittab** entry using *uugetty* on an intelligent modem or direct line:

30:2:respawn:/usr/lib/uucp/uugetty -r -t 60 tty12 1200

FILES

/etc/gettydefs
/etc/issue

SEE ALSO

uucico(1M), getty(1M), init(1M), tty(7).
ct(1C), cu(1C), login(1) in the *AT&T 3B2 Computer User Reference Manual.*
ioctl(2), gettydefs(4), inittab(4) in the *AT&T 3B2 Computer Programmer Reference Manual.*

BUGS

Ct will not work when uugetty is used with an intelligent modem such as penril or ventel.

NAME

uusched − the scheduler for the uucp file transport program

SYNOPSIS

/usr/lib/uucp/uusched [−x debug_level] [−u debug_level]

DESCRIPTION

Uusched is the *uucp* file transport scheduler. It is usually started by the daemon *uudemon.hour* that is started by *cron*;

39 * * * * /bin/su uucp -c "/usr/lib/uucp/uudemon.hour > /dev/null"

The two options are for debugging purposes only; −x *debug_level* will output debugging messages from *uusched* and −u *debug_level* will be passed as −x *debug_level* to *uucico*. The *debug_level* is a number between 0 and 9; higher numbers give more detailed information. Note, however, that compiling *uucp* with the -DSMALL option will result in little debugging output.

FILES

/usr/lib/uucp/Systems
/usr/lib/uucp/Permissions
/usr/lib/uucp/Devices
/usr/spool/uucp/*
/usr/spool/locks/LCK*
/usr/spool/uucppublic/*

SEE ALSO

cron(1M), uucico(1M).
uucp(1C), uustat(1C), uux(1C) in the *AT&T 3B2 Computer User Referene Manual.*

NAME

Uutry – try to contact remote system with debugging on

SYNOPSIS

/usr/lib/uucp/Uutry [**–x** debug_level] [**–r**] system_name

DESCRIPTION

Uutry is a shell that is used to invoke *uucico* to call a remote site. Debugging is turned on (default is level 5); **–x** will override that value. The **–r** overrides the retry time in **/usr/spool/uucp/.status**. The debugging output is put in file **/tmp/***system_name*. A tail **–f** of the output is executed. A <RUBOUT> or <BREAK> will give control back to the terminal while the *uucico* continues to run, putting its output in **/tmp/***system_name*.

FILES

/usr/lib/uucp/Systems
/usr/lib/uucp/Permissions
/usr/lib/uucp/Devices
/usr/lib/uucp/Maxuuxqts
/usr/lib/uucp/Maxuuscheds
/usr/spool/uucp/*
/usr/spool/locks/LCK*
/usr/spool/uucppublic/*
/tmp/system_name

SEE ALSO

uucico(1M).
uucp(1C), uux(1C) in the *AT&T 3B2 Computer User Reference Manual.*

NAME

uuxqt − execute remote command requests

SYNOPSIS

/usr/lib/uucp/uuxqt [−s system] [−x debug_level]

DESCRIPTION

Uuxqt is the program that executes remote job requests from remote systems generated by the use of the *uux* command. (*Mail* uses *uux* for remote mail requests). *Uuxqt* searches the spool directories looking for *X.* files. For each *X.* file, *uuxqt* checks to see if all the required data files are available and accessible, and file commands are permitted for the requesting system. The *Permissions* file is used to validate file accessibility and command execution permission.

There are two environment variables that are set before the *uuxqt* command is executed:
UU_MACHINE is the machine that sent the job (the previous one).
UU_USER is the user that sent the job.
These can be used in writing commands that remote systems can execute to provide information, auditing, or restrictions.

The -x *debug_level* is a single digit between 0 and 9. Higher numbers give more detailed debugging information. Note, however, that compiling *uucp* with the −**DSMALL** option will result in little debugging output.

FILES

/usr/lib/uucp/Permissions
/usr/lib/uucp/Maxuuxqts
/usr/spool/uucp/*
/usr/spool/locks/LCK*

SEE ALSO

uucico(1M).
mail(1), uucp(1C), uustat(1C), uux(1C) in the *AT&T 3B2 Computer User Reference Manual.*

NAME

volcopy – make literal copy of file system

SYNOPSIS

/etc/volcopy [options] fsname srcdevice volname1 destdevice volname2

DESCRIPTION

Volcopy makes a literal copy of the file system using a blocksize matched to the device. *Options* are:

–a invoke a verification sequence requiring a positive operator response instead of the standard 10 second delay before the copy is made

–s (default) invoke the **DEL if wrong** verification sequence.

Other *options* are used only with reel tapes (not applicable to the 3B2 computer cartridge tape):

–bpidensity bits-per-inch (i.e., **800/1600/6250**),

–feetsize size of reel in feet (i.e., **1200/2400**),

–reelnum beginning reel number for a restarted copy,

–buf use double buffered I/O.

The program requests length and density information if it is not given on the command line or is not recorded on an input tape label. If the file system is too large to fit on one reel, *volcopy* will prompt for additional reels. Labels of all reels are checked. Tapes may be mounted alternately on two or more drives. If *volcopy* is interrupted, it will ask if the user wants to quit or wants a shell. In the latter case, the user can perform other operations (e.g., *labelit*) and return to *volcopy* by exiting the new shell.

The *fsname* argument represents the mounted name (e.g.: **root**, **u1**, etc.) of the filsystem being copied.

The *srcdevice* or *destdevice* should be the physical disk section or tape (e.g.: **/dev/dsk/cld0s8**, **/dev/rdsk/cld1s8**, etc.).

The *volname* is the physical volume name (e.g.: **pk3**, **t0122**, etc.) and should match the external label sticker. Such label names are limited to six or fewer characters. *Volname* may be – to use the existing volume name.

Srcdevice and *volname1* are the device and volume from which the copy of the file system is being extracted. *Destdevice* and *volname2* are the target device and volume.

Fsname and *volname* are recorded in the last 12 characters of the superblock (**char fsname[6], volname[6];**).

FILES

/etc/log/filesave.log a record of file systems/volumes copied

SEE ALSO

labelit(1M).
fs(4) in the *AT&T 3B2 Computer Programmer Reference Manual.*
sh(1) in the *AT&T 3B2 Computer User Reference Manual.*

BUGS

Only device names beginning **/dev/rmt/** are treated as tapes.

NAME

whodo – who is doing what

SYNOPSIS

/etc/whodo

DESCRIPTION

Whodo produces merged, reformatted, and dated output from the *who*(1) and *ps*(1) commands.

FILES

etc/passwd

SEE ALSO

ps(1), who(1) in the *AT&T 3B2 Computer User Reference Manual.*

Replace this

page with the

Section 7 (Special Files)

tab separator.

NAME

intro – introduction to special files

DESCRIPTION

This section describes various special files that refer to specific hardware peripherals and UNIX system device drivers. The names of the entries are generally derived from names for the hardware, as opposed to the names of the special files themselves. Characteristics of both the hardware device and the corresponding UNIX system device driver are discussed where applicable.

Disk device file names are in the following format:

/dev/{r}dsk/c#d#s#

where **r** indicates a raw interface to the disk, the **c#** indicates the controller number, **d#** indicates the device attached to the controller and **s#** indicates the section number of the partitioned device.

SEE ALSO

Disk Partitioning in the *3B2 Computer System Administration Utilities Guide*.

NAME

console - console interface

DESCRIPTION

The console provides the operator interface to the 3B2 Computer.

The file **/dev/console** refers to the system console. This special file implements the features described in termio(7).

The file **/dev/contty** refers to an asynchronous serial data line originating from the system board. This special file implements the features described in termio(7).

FILES

/dev/console
/dev/contty

SEE ALSO

termio(7).

NAME

hdelog – hard disk error log interface

DESCRIPTION

Hdelog is a special file that provides access to the disk error logging mechanism, the equipped disk table, and the disk drivers of the equipped disks for doing physical (non-partitioned) disk I/O. It is an internal interface of bad block handling and a few other disk utilities and is not intended to be used directly by users. You must be super-user to use it. This command is a bad block handling utility command. You must be super-user to use it.

FILES

/dev/hdelog

SEE ALSO

hdeadd(1M), hdefix(1M), hdelogger(1M).

NAME

id – 3B2 Computer Integral Disk Subsystem

DESCRIPTION

The 3B2 Computer integral disk subsystem consists of two types of drives.. They are Seagate and CDC. Both types of drives contain only fixed media. The Seagate media contains 21888 blocks and the CDC media contains 62550 blocks. A 3B2 Computer may contain only one drive. The files **idsk00**, ..., **idsk07** refer to sections of the drive unit number 0. This slicing allows the media to be broken up into more manageable pieces.

The origin and size of the sections on each drive are as follows:

/* For Seagate drives */

SIZE	START CYL.	PARTITION		
20808	16	partition 0 -	cyl 16-304	(root fs)
13248	121	partition 1 -	cyl 121-304	(swap)
9720	170	partition 2 -	cyl 170-304	(usr)
7200	205	partition 3 -	cyl 205-304	
3600	255	partition 4 -	cyl 255-304	
21816	2	partition 5 -	cyl 2-304	(init fs)
21888	1	partition 6 -	cyl 1-304	(full disk)
72	1	partition 7 -	cyl 1	(boot)

/* For CDC drives */

SIZE	START CYL.	PARTITION		
20808	14	partition 0 -	cyl 14-695	(root fs)
13248	124	partition 1 -	cyl 124-695	(swap)
9720	170	partition 2 -	cyl 170-695	(usr)
7200	345	partition 3 -	cyl 345-695	
3600	520	partition 4 -	cyl 520-695	
21816	2	partition 5 -	cyl 2-695	(init fs)
21888	1	partition 6 -	cyl 1-695	(full disk)
72	1	partition 7 -	cyl 1	(boot)

The start address is a cylinder address, with length representing the number of blocks within the section. Also it should be noted that **idsk00 has been reserved for the idsk07** has been reserved to contain the system boot blocks.

The **ifdsk** files access the disk via 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 disk files begin with **ridsk** and end with a number which selects the same disk section as the corresponding **idsk** file.

In raw I/O the buffer must begin on a word boundary, and transfer counts can be as small as single byte.

FILES

/dev/idsk*, /dev/ridsk*

NAME

if – 3B2 Computer Floppy Disk Subsystem

DESCRIPTION

The 3B2 Computer floppy disk subsystem consists of one Tandon Floppy drive. The media contains 1422 blocks. The files **ifdsk00**, ..., **ifdsk07** refer to sections of the floppy disk drive. This slicing allows the media to be broken up into more manageable pieces.

The origin and size of the sections on each drive are as follows:

SIZE	START CYL.	PARTITION
1422	0	/* partition 0 track...0 */
1404	1	/* partition 1 track...1-78 */
1170	14	/* partition 2 track...14-78 */
936	27	/* partition 3 track...27-78 */
702	40	/* partition 4 track...0-78 */
468	53	/* partition 5 track...0-78 */
234	66	/* partition 6 track...0-78 */
18	0	/* partition 7 track...0-1..BOOT */

The start address is a cylinder address, with length representing the number of blocks within the section.

The **ifdsk** files access the disk via 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 disk files begin with **rifdsk** and end with a number which selects the same disk section as the corresponding **ifdsk** file.

In raw I/O the buffer must begin on a word boundary, and transfer counts can be as small as single byte.

FILES

/dev/ifdsk*, /dev/rifdsk*

NAME

mem, kmem — core memory

DESCRIPTION

Mem is a special file that is an image of the core memory of the computer. It may be used, for example, to examine, and even to patch the system.

Byte addresses in *mem* are interpreted as memory addresses. References to non-existent locations cause errors to be returned.

Examining and patching device registers is likely to lead to unexpected results when read-only or write-only bits are present.

The file *kmem* is the same as *mem* except that kernel virtual memory rather than physical memory is accessed.

The I/O addresses begin at location 0x60000 of *kmem* and per-process data for the current process begins at 0x80880000.

FILES

/dev/mem
/dev/kmem

BUGS

Some of *kmem* cannot be read because of write-only addresses or unequipped memory addresses.

NAME

mt – tape interface

DESCRIPTION

The files **mt/ctape?** and **rmt/ctape?** refer to cartridge tape controllers (CTC) and associated tape drives. These special device files are linked to the standard CTC **SA/ctape?** and **rSA/ctape?** files, respectively.

The *finc*(1M), *frec*(1M), and *labelit*(1M) commands require these magnetic tape file names to work correctly with the CTC. No other CTC commands require these file names.

FILES

/dev/mt/ctape*
/dev/rmt/ctape*

SEE ALSO

finc(1M), frec(1M), labelit(1M).

NAME

null — the null file

DESCRIPTION

Data written on a null special file is discarded.

Reads from a null special file always return 0 bytes.

FILES

/dev/null

NAME

port - 5 line asynchronous interface

DESCRIPTION

Each of the five lines attached to a port behaves as described in *termio*(7). Each port supports 4 RS232 lines and one parallel Centronics interface. The **c_cflag** of B50, B75, B200, EXTA, and EXTB are not available.

FILES

/dev/tty?? serial interface
/dev/lp? parallel interface

SEE ALSO

termio(7).

NAME

prf — operating system profiler

DESCRIPTION

The file **prf** provides access to activity information in the operating system. Writing the file loads the measurement facility with text of addresses to be monitored. Reading the file returns these addresses and a set of counters indicative of activity between adjacent text addresses.

The recording mechanism is driven by the system clock and samples the program counter at line frequency. Samples that catch the operating system are matched against the stored text addresses and increment corresponding counters for later processing.

The file **prf** is a psuedo-device with no associated hardware.

FILES

/dev/prf

SEE ALSO

config(1M), profiler(1M).

NAME

SA – devices administered by Simple Administration

DESCRIPTION

The files in the directories **/dev/SA** (for block devices) and the **/dev/rSA** (for raw devices) are used by Simple Administration to access the devices on which it operates. For devices that support more than one partition (like disks) the **/dev/(r)SA** entry is linked to the partition that spans the entire device. Not all **/dev/(r)SA** entries are used by all Simple Administration commands.

FILES

/dev/SA
/dev/rSA

SEE ALSO

sysadm(1) in the *AT&T 3B2 Computer User Reference Manual.*

NAME

sxt – pseudo-device driver

DESCRIPTION

Sxt is a pseudo-device driver that interposes a discipline between the standard *tty* line disciplines and a real device driver. The standard disciplines manipulate *virtual tty* structures (channels) declared by the *sxt* driver. *Sxt* acts as a discipline manipulating a *real tty* structure declared by a real device driver. The *sxt* driver is currently only used by the *shl*(1) command.

Virtual ttys are named by inodes in the subdirectory **/dev/sxt** and are allocated in groups of up to eight. To allocate a group, a program should exclusively open a file with a name of the form **/dev/sxt/??0** (channel 0) and then execute a SXTIOCLINK *ioctl* call to initiate the multiplexing.

Only one channel, the *controlling* channel, can receive input from the keyboard at a time; others attempting to read will be blocked.

There are two groups of *ioctl*(2) commands supported by *sxt*. The first group contains the standard *ioctl* commands described in *termio*(7), with the addition of the following:

TIOCEXCL Set *exclusive use* mode: no further opens are permitted until the file has been closed.

TIOCNXCL Reset *exclusive use* mode: further opens are once again permitted.

The second group are directives to *sxt* itself. Some of these may only be executed on channel 0.

SXTIOCLINK Allocate a channel group and multiplex the virtual ttys onto the real tty. The argument is the number of channels to allocate. This command may only be executed on channel 0. Possible errors include:

- EINVAL The argument is out of range.
- ENOTTY The command was not issued from a real tty.
- ENXIO *linesw* is not configured with *sxt*.
- EBUSY An SXTIOCLINK command has already been issued for this real *tty*.
- ENOMEM There is no system memory available for allocating the virtual tty structures.
- EBADF Channel 0 was not opened before this call.

SXTIOCSWTCH Set the controlling channel. Possible errors include:

- EINVAL An invalid channel number was given.
- EPERM The command was not executed from channel 0.

SXTIOCWF Cause a channel to wait until it is the controlling channel. This command will return the error, *EINVAL*, if an invalid channel number is given.

SXTIOCUBLK	Turn off the **loblk** control flag in the virtual tty of the indicated channel. The error *EINVAL* will be returned if an invalid number or channel 0 is given.
SXTIOCSTAT	Get the status (blocked on input or output) of each channel and store in the *sxtblock* structure referenced by the argument. The error *EFAULT* will be returned if the structure cannot be written.
SXTIOCTRACE	Enable tracing. Tracing information is written to the console on the 3B2 Computer. This command has no effect if tracing is not configured.
SXTIOCNOTRACE	Disable tracing. This command has no effect if tracing is not configured.

FILES

/dev/sxt/??[0-7]	Virtual tty devices

SEE ALSO

termio(7).
shl(1), stty(1) in the *AT&T 3B2 Computer User Reference Manual.*
ioctl(2), open(2) in the *AT&T 3B2 Computer Programmer Reference Manual.*

NAME

termio – general terminal interface

DESCRIPTION

All of the asynchronous communications ports use the same general interface, no matter what hardware is involved. The remainder of this section discusses the common features of this interface.

When a terminal file is opened, it normally causes the process to wait until a connection is established. In practice, user's programs seldom open these files; they are opened by *getty* and become a user's standard input, output, and error files. The very first terminal file opened by the process group leader of a terminal file not already associated with a process group becomes the *control terminal* for that process group. The control terminal plays a special role in handling quit and interrupt signals, as discussed below. The control terminal is inherited by a child process during a *fork*(2). A process can break this association by changing its process group using *setpgrp*(2).

A terminal associated with one of these files ordinarily operates in full-duplex mode. Characters may be typed at any time, even while output is occurring, and are only lost when the system's character input buffers become completely full, which is rare, or when the user has accumulated the maximum allowed number of input characters that have not yet been read by some program. Currently, this limit is 256 characters. When the input limit is reached, all the saved characters are thrown away without notice.

Normally, terminal input is processed in units of lines. A line is delimited by a new-line (ASCII LF) character, an end-of-file (ASCII EOT) character, or an end-of-line character. This means that a program attempting to read will be suspended until an entire line has been typed. Also, no matter how many characters are requested in the read call, at most one line will be returned. It is not, however, necessary to read a whole line at once; any number of characters may be requested in a read, even one, without losing information.

During input, erase and kill processing is normally done. By default, the character # erases the last character typed, except that it will not erase beyond the beginning of the line. By default, the character @ kills (deletes) the entire input line, and optionally outputs a new-line character. Both these characters operate on a keystroke basis, independently of any backspacing or tabbing that may have been done. Both the erase and kill characters may be entered literally by preceding them with the escape character (\). In this case the escape character is not read. The erase and kill characters may be changed.

Certain characters have special functions on input. These functions and their default character values are summarized as follows:

INTR (Rubout or ASCII DEL) generates an *interrupt* signal which is sent to all processes with the associated control terminal. Normally, each such process is forced to terminate, but arrangements may be made either to ignore the signal or to receive a trap to an agreed-upon location; see *signal*(2).

QUIT (Control-| or ASCII FS) generates a *quit* signal. Its treatment is identical to the interrupt signal except that, unless a receiving process has made other arrangements, it will not only be terminated but a core image file (called **core**) will be created in the current working directory.

SWTCH (Control-z or ASCII SUB) is used by the job control facility, *shl*, to change the current layer to the control layer.

ERASE (#) erases the preceding character. It will not erase beyond the start of a line, as delimited by a NL, EOF, or EOL character.

KILL (@) deletes the entire line, as delimited by a NL, EOF, or EOL character.

EOF (Control-d or ASCII EOT) may be used to generate an end-of-file from a terminal. When received, all the characters waiting to be read are immediately passed to the program, without waiting for a new-line, and the EOF is discarded. Thus, if there are no characters waiting, which is to say the EOF occurred at the beginning of a line, zero characters will be passed back, which is the standard end-of-file indication.

NL (ASCII LF) is the normal line delimiter. It can not be changed or escaped.

EOL (ASCII NUL) is an additional line delimiter, like NL. It is not normally used.

STOP (Control-s or ASCII DC3) can be used to temporarily suspend output. It is useful with CRT terminals to prevent output from disappearing before it can be read. While output is suspended, STOP characters are ignored and not read.

START (Control-q or ASCII DC1) is used to resume output which has been suspended by a STOP character. While output is not suspended, START characters are ignored and not read. The start/stop characters can not be changed or escaped.

The character values for INTR, QUIT, SWTCH, ERASE, KILL, EOF, and EOL may be changed to suit individual tastes. The ERASE, KILL, and EOF characters may be escaped by a preceding \ character, in which case no special function is done.

When the carrier signal from the data-set drops, a *hang-up* signal is sent to all processes that have this terminal as the control terminal. Unless other arrangements have been made, this signal causes the processes to terminate. If the hang-up signal is ignored, any subsequent read returns with an end-of-file indication. Thus, programs that read a terminal and test for end-of-file can terminate appropriately when hung up on.

When one or more characters are written, they are transmitted to the terminal as soon as previously-written characters have finished typing. Input characters are echoed by putting them in the output queue as they arrive. If a process produces characters more rapidly than they can be typed, it will be suspended when its output queue exceeds some limit. When the queue has drained down to some threshold, the program is resumed.

Several *ioctl*(2) system calls apply to terminal files. The primary calls use the following structure, defined in **<termio.h>**:

```
#define NCC 8
struct termio {
        unsigned short c_iflag;     /* input modes */
        unsigned short c_oflag;     /* output modes */
        unsigned short c_cflag;     /* control modes */
        unsigned short c_lflag;     /* local modes */
        char           c_line;      /* line discipline */
        unsigned char  c_cc[NCC];   /* control chars */
```

};

The special control characters are defined by the array *c_cc*. The relative positions and initial values for each function are as follows:

0	VINTR	DEL
1	VQUIT	FS
2	VERASE	#
3	VKILL	@
4	VEOF	EOT
5	VEOL	NUL
6	reserved	
7	SWTCH	

The *c_iflag* field describes the basic terminal input control:

IGNBRK	0000001	Ignore break condition.
BRKINT	0000002	Signal interrupt on break.
IGNPAR	0000004	Ignore characters with parity errors.
PARMRK	0000010	Mark parity errors.
INPCK	0000020	Enable input parity check.
ISTRIP	0000040	Strip character.
INLCR	0000100	Map NL to CR on input.
IGNCR	0000200	Ignore CR.
ICRNL	0000400	Map CR to NL on input.
IUCLC	0001000	Map upper-case to lower-case on input.
IXON	0002000	Enable start/stop output control.
IXANY	0004000	Enable any character to restart output.
IXOFF	0010000	Enable start/stop input control.

If IGNBRK is set, the break condition (a character framing error with data all zeros) is ignored, that is, not put on the input queue and therefore not read by any process. Otherwise if BRKINT is set, the break condition will generate an interrupt signal and flush both the input and output queues. If IGNPAR is set, characters with other framing and parity errors are ignored.

If PARMRK is set, a character with a framing or parity error which is not ignored is read as the three-character sequence: 0377, 0, X, where X is the data of the character received in error. To avoid ambiguity in this case, if ISTRIP is not set, a valid character of 0377 is read as 0377, 0377. If PARMRK is not set, a framing or parity error which is not ignored is read as the character NUL (0).

If INPCK is set, input parity checking is enabled. If INPCK is not set, input parity checking is disabled. This allows output parity generation without input parity errors.

If ISTRIP is set, valid input characters are first stripped to 7-bits, otherwise all 8-bits are processed.

If INLCR is set, a received NL character is translated into a CR character. If IGNCR is set, a received CR character is ignored (not read). Otherwise if ICRNL is set, a received CR character is translated into a NL character.

If IUCLC is set, a received upper-case alphabetic character is translated into the corresponding lower-case character.

If IXON is set, start/stop output control is enabled. A received STOP character will suspend output and a received START character will restart output. All start/stop characters are ignored and not read. If IXANY is set, any input character, will restart output which has been suspended.

If IXOFF is set, the system will transmit START/STOP characters when the input queue is nearly empty/full.

The initial input control value is all-bits-clear.

The *c_oflag* field specifies the system treatment of output:

OPOST	0000001	Postprocess output.
OLCUC	0000002	Map lower case to upper on output.
ONLCR	0000004	Map NL to CR-NL on output.
OCRNL	0000010	Map CR to NL on output.
ONOCR	0000020	No CR output at column 0.
ONLRET	0000040	NL performs CR function.
OFILL	0000100	Use fill characters for delay.
OFDEL	0000200	Fill is DEL, else NUL.
NLDLY	0000400	Select new-line delays:
NL0	0	
NL1	0000400	
CRDLY	0003000	Select carriage-return delays:
CR0	0	
CR1	0001000	
CR2	0002000	
CR3	0003000	
TABDLY	0014000	Select horizontal-tab delays:
TAB0	0	
TAB1	0004000	
TAB2	0010000	
TAB3	0014000	Expand tabs to spaces.
BSDLY	0020000	Select backspace delays:
BS0	0	
BS1	0020000	
VTDLY	0040000	Select vertical-tab delays:
VT0	0	
VT1	0040000	
FFDLY	0100000	Select form-feed delays:
FF0	0	
FF1	0100000	

If OPOST is set, output characters are post-processed as indicated by the remaining flags, otherwise characters are transmitted without change.

If OLCUC is set, a lower-case alphabetic character is transmitted as the corresponding upper-case character. This function is often used in conjunction with IUCLC.

If ONLCR is set, the NL character is transmitted as the CR-NL character pair. If OCRNL is set, the CR character is transmitted as the NL character. If ONOCR is set, no CR character is transmitted when at column 0 (first position). If ONLRET is set, the NL character is assumed to do the carriage-return function; the column pointer will be set to 0 and the delays specified for CR will be used. Otherwise the NL character is assumed to do just the line-feed function; the column pointer will remain unchanged. The column pointer is also set to 0 if the CR character is actually transmitted.

The delay bits specify how long transmission stops to allow for mechanical or other movement when certain characters are sent to the terminal. In all cases a value of 0 indicates no delay. If OFILL is set, fill characters will be transmitted for delay instead of a timed delay. This is useful for high baud rate terminals which need

only a minimal delay. If OFDEL is set, the fill character is DEL, otherwise NUL.

If a form-feed or vertical-tab delay is specified, it lasts for about 2 seconds.

New-line delay lasts about 0.10 seconds. If ONLRET is set, the carriage-return delays are used instead of the new-line delays. If OFILL is set, two fill characters will be transmitted.

Carriage-return delay type 1 is dependent on the current column position, type 2 is about 0.10 seconds, and type 3 is about 0.15 seconds. If OFILL is set, delay type 1 transmits two fill characters, and type 2, four fill characters.

Horizontal-tab delay type 1 is dependent on the current column position. Type 2 is about 0.10 seconds. Type 3 specifies that tabs are to be expanded into spaces. If OFILL is set, two fill characters will be transmitted for any delay.

Backspace delay lasts about 0.05 seconds. If OFILL is set, one fill character will be transmitted.

The actual delays depend on line speed and system load.

The initial output control value is all bits clear.

The *c_cflag* field describes the hardware control of the terminal:

CBAUD	0000017	Baud rate:
B0	0	Hang up
B600	0000010	600 baud
B1200	0000011	1200 baud
B1800	0000012	1800 baud
B2400	0000013	2400 baud
B4800	0000014	4800 baud
B9600	0000015	9600 baud
EXTA	0000016	External A
EXTB	0000017	External B
CSIZE	0000060	Character size:
CS5	0	5 bits
CS6	0000020	6 bits
CS7	0000040	7 bits
CS8	0000060	8 bits
CSTOPB	0000100	Send two stop bits, else one.
CREAD	0000200	Enable receiver.
PARENB	0000400	Parity enable.
PARODD	0001000	Odd parity, else even.
HUPCL	0002000	Hang up on last close.
CLOCAL	0004000	Local line, else dial-up.
LOBLK	0010000	Block layer output.

The CBAUD bits specify the baud rate. The zero baud rate, B0, is used to hang up the connection. If B0 is specified, the data-terminal-ready signal will not be asserted. Normally, this will disconnect the line. For any particular hardware, impossible speed changes are ignored.

The CSIZE bits specify the character size in bits for both transmission and reception. This size does not include the parity bit, if any. If CSTOPB is set, two stop bits are used, otherwise one stop bit. For example, at 110 baud, two stops bits are required.

If PARENB is set, parity generation and detection is enabled and a parity bit is added to each character. If parity is enabled, the PARODD flag specifies odd parity

if set, otherwise even parity is used.

If CREAD is set, the receiver is enabled. Otherwise no characters will be received.

If HUPCL is set, the line will be disconnected when the last process with the line open closes it or terminates. That is, the data-terminal-ready signal will not be asserted.

If CLOCAL is set, the line is assumed to be a local, direct connection with no modem control. Otherwise modem control is assumed.

If LOBLK is set, the output of a job control layer will be blocked when it is not the current layer. Otherwise the output generated by that layer will be multiplexed onto the current layer.

The initial hardware control value after open is B300, CS8, CREAD, HUPCL.

The *c_lflag* field of the argument structure is used by the line discipline to control terminal functions. The basic line discipline (0) provides the following:

ISIG	0000001	Enable signals.
ICANON	0000002	Canonical input (erase and kill processing).
XCASE	0000004	Canonical upper/lower presentation.
ECHO	0000010	Enable echo.
ECHOE	0000020	Echo erase character as BS-SP-BS.
ECHOK	0000040	Echo NL after kill character.
ECHONL	0000100	Echo NL.
NOFLSH	0000200	Disable flush after interrupt or quit.

If ISIG is set, each input character is checked against the special control characters INTR, SWTCH, and QUIT. If an input character matches one of these control characters, the function associated with that character is performed. If ISIG is not set, no checking is done. Thus these special input functions are possible only if ISIG is set. These functions may be disabled individually by changing the value of the control character to an unlikely or impossible value (e.g., 0377).

If ICANON is set, canonical processing is enabled. This enables the erase and kill edit functions, and the assembly of input characters into lines delimited by NL, EOF, and EOL. If ICANON is not set, read requests are satisfied directly from the input queue. A read will not be satisfied until at least MIN characters have been received or the timeout value TIME has expired between characters. This allows fast bursts of input to be read efficiently while still allowing single character input. The MIN and TIME values are stored in the position for the EOF and EOL characters, respectively. The time value represents tenths of seconds.

If XCASE is set, and if ICANON is set, an upper-case letter is accepted on input by preceding it with a \ character, and is output preceded by a \ character. In this mode, the following escape sequences are generated on output and accepted on input:

for:	*use*:
`	\'
|	\!
~	\^
{	\(
}	\)
\	\\

For example, A is input as \a, \n as \\n, and \N as \\\n.

If ECHO is set, characters are echoed as received.

When ICANON is set, the following echo functions are possible. If ECHO and ECHOE are set, the erase character is echoed as ASCII BS SP BS, which will clear the last character from a CRT screen. If ECHOE is set and ECHO is not set, the erase character is echoed as ASCII SP BS. If ECHOK is set, the NL character will be echoed after the kill character to emphasize that the line will be deleted. Note that an escape character preceding the erase or kill character removes any special function. If ECHONL is set, the NL character will be echoed even if ECHO is not set. This is useful for terminals set to local echo (so-called half duplex). Unless escaped, the EOF character is not echoed. Because EOT is the default EOF character, this prevents terminals that respond to EOT from hanging up.

If NOFLSH is set, the normal flush of the input and output queues associated with the quit, switch, and interrupt characters will not be done.

The initial line-discipline control value is all bits clear.

The primary *ioctl*(2) system calls have the form:

```
ioctl (fildes, command, arg)
struct termio *arg;
```

The commands using this form are:

TCGETA	Get the parameters associated with the terminal and store in the *termio* structure referenced by **arg**.
TCSETA	Set the parameters associated with the terminal from the structure referenced by **arg**. The change is immediate.
TCSETAW	Wait for the output to drain before setting the new parameters. This form should be used when changing parameters that will affect output.
TCSETAF	Wait for the output to drain, then flush the input queue and set the new parameters.

Additional *ioctl*(2) calls have the form:

```
ioctl (fildes, command, arg)
int arg;
```

The commands using this form are:

TCSBRK	Wait for the output to drain. If *arg* is 0, then send a break (zero bits for 0.25 seconds).
TCXONC	Start/stop control. If *arg* is 0, suspend output; if 1, restart suspended output.
TCFLSH	If *arg* is 0, flush the input queue; if 1, flush the output queue; if 2, flush both the input and output queues.

FILES

/dev/tty*

SEE ALSO

stty(1) in the *AT&T 3B2 Computer User Reference Manual.*
fork(2), ioctl(2), setpgrp(2), signal(2) in the *AT&T 3B2 Computer Programmer Reference Manual.*

NAME

tty — controlling terminal interface

DESCRIPTION

The file **/dev/tty** is, in each process, a synonym for the control terminal associated with the process group of that process, if any. It is useful for programs or shell sequences that wish to be sure of writing messages on the terminal no matter how output has been redirected. It can also be used for programs that demand the name of a file for output, when typed output is desired and it is tiresome to find out what terminal is currently in use.

FILES

/dev/tty
/dev/tty*

SEE ALSO

console(7), ports(7).

Replace this

page with the

Section 8 (Maint. Procedures)

tab separator.

NAME

intro – introduction to system maintenance procedures

DESCRIPTION

This section outlines certain procedures that will be of interest to those charged with the task of system maintenance. Included are discussions on such topics as boot procedures, recovery from crashes, file backups, etc.

BUGS

No manual can take the place of good, solid experience.

NAME

3B2boot – 3B2 Computer bootstrap procedures

DESCRIPTION

The bootstrap procedure on the 3B2 Computer consists of three phases. The first phase is initiated by the user typing boot while in firmware. This phase will prompt for the name of the file to be booted, then the device from which to boot. (This phase is also initiated after power-up, and upon completion of diagnostics. In this case, the file to boot defaults to unix and the device to boot defaults to id.) The second phase is the execution of *mboot* which reads from the boot block of the selected boot device. *Mboot* is loaded at physical address 0x2004000. Its function is to read the *lboot* phase from the remainder of the boot partition into the top of the first half megabyte of main memory and to initiate *lboot* execution.

Any error during the *mboot* phase will cause a return to the firmware.

The *lboot* phase has been passed the name of the file to be booted. If the name given is a directory on the init file system, *lboot* will respond with a listing of the files present in that directory. Once a valid file name has been obtained, *lboot* loads the file into memory and starts the execution of the file.

SEE ALSO

newboot(1M).
a.out(4) in the *AT&T 3B2 Computer Programmer Reference Manual.*

DIAGNOSTICS

Self-explanatory messages about bad directory entries and bad file formats.

BUGS

Lboot isn't smart enough to know which *a.out* files can be used as bootable programs. If an *a.out* is specified that is not bootable, *lboot* will load it and branch to it. What happens after that is unpredictable.

NAME

baud - display or change console baud rate

SYNOPSIS

baud

q

DESCRIPTION

Baud is a firmware command that displays the console baud rate, then prompts for a new baud rate. It is not available in all ROM code issues. The default entry, <CR> will not change the console baud rate, otherwise the system will change the console's baud rate to the requested value and save the value in NVRAM, provided it is a supported rate. *Baud* complains about unsupported baud rates. *Baud* is available in ROM code with an issue number greater that or equal to 0x0c (12).

Q(uit) provides a return to the Maintenance Control Program (MCP) null mode (or the Debug Monitor (DEMON), if equipped).

SEE ALSO

version(8)

NAME

dgmon – diagnostic monitor

SYNOPSIS

dgmon

Dgmon Sub-commands

dgn
dgn [unit[=number]]
dgn [unit[=number]] [ph=a[-b]] [rep=n] [ucl] [soak]
l(ist) unit
h(elp)
s(how)

(phase range specified ph=a-b means phases a through b)

DESCRIPTION

The diagnostic monitor is a firmware-level utility that runs diagnostics on the 3B2 Computer via the system console. The interface and the entry into the diagnostic mode is discussed in detail in the *AT&T 3B2 Computer System Administration Utilities* Diagnostics can be invoked for the entire computer, types of devices (e.g., all ports boards), a specific device (e.g., the system board), or a particular phase or range of phases for the device or device type. Each diagnostic phase and phase description can be found by listing the diagnostic phases for each computer device (eg. 1 sbd). The Monitor will list the 3B2 Computer's Equipped Device Table (EDT) with the *s(how)* command.

Types of diagnostic phases:

Normal-Diagnostics that run every time the computer is powered up

Demand-Diagnostics that run during soak or must be specifically requested

Interactive-Diagnostics that must be specifically requested and may cause loss of stored data or require operator intervention.

(Diagnostic Monitor will deny request for diagnostics on unequipped devices and non-existent phases)

Option definitions:

ucl -unconditional execution, run all specified phases and display all failing results

rep=n -repeat phases(s) n times

ph=a -select individual phase

ph=a-b -select phase range a through b

soak -silently and continuously run normal and demand diagnostics for specified range (default: all of phase table) and for specified repititions (default: continuous-stopped with keyboard entry)

EXAMPLES

dgn (full system)

dgn ports (all ports devices)

```
dgn sbd 0 ucl       (Unconditional execution)

dgn ports 0         (ports 0 diagnostic)

dgn ports 1 ucl

dgn sbd ph=3        (phase specification)

dgn sbd ph=1-5      (phase range specification)

dgn sbd soak        (diagnostic system board soak)
```

Whenever specific phases are requested, the device to be tested must be designated.

FILES

/dgmon

DIAGNOSTICS

Improperly typed syntax will generate message or specify the invalid input. The "h" command will generate a listing of the correct syntax for the system board firmware.

NAME

edittbl – edit edt_data file

SYNOPSIS

edittbl **–d|s** [**–g**] [**–i**] [**–l**] [**–r**] [**–t**] [file]

DESCRIPTION

Edittbl is a user level utility that permits changes to **edt_data**, the file in the root file system **FILLEDT**. **Edt_data** is read during the Equipped Device Table (EDT) completion to get the device and subdevice look-up tables. This utility permits independent selection of device or subdevice tables, generation of either base table, new entry installation for either table, entry removal for the device table, and entry listings for either or both tables.

Edittbl prints the option list if the command has no arguments. The arguments are:

–d This option selects the device look-up table for the utility's operation(s).

–s This option selects the subdevice look-up table for the utility's operation(s).

–g This option will generate the base look-up table entries for the selected look-up table(s). For the device table, these base entries are SBD, and PORTS. For the subdevice table, they are NULL, FD5, HD10, and HD30.

–i This option specifies that new entries are to be added to the selected table. The ID codes for table entries and the input are compared; only new codes are installed. The formats for entries are described below. An EOF or "." end the data input.

–l This option specifies that the selected table(s) are listed.

–r This option specifies that entries are to be removed from the device look-up table. When removing subdevice look-up table entries from **init/dgn/edt_data** conjunction with removing a device entry, this utility will check the Equipped Device Table (EDT) to verify that no subdevices specified for removal are present. The ID codes of the table are compared to the input and entries are removed for matches. The format is identical to that for the **–i** option and is listed below. An EOF or "." end the data input.

–t This option suppresses the program headings and user prompts; warnings and errors are not affected. This option is primarily useful in installation and removal scripts.

file The user may specify a target path name for the utilities. If none is specified, **./edt_data** is the default.

INPUT FORMAT

Data for installation/removal are entered as hex format numbers or character strings, one line for each table entry. The data fields must be supplied in the sequence described.

Devices

ID code This field is a number between 0x0 and 0xffff that a device uses to identify itself. ID codes are administered by AT&T Technologies.

name This field is a character string (maximum of 9 characters) that holds the user-recognizable name for a device. Device names are administered by AT&T Technologies. This string is also the file name that DGMON loads to diagnose a device.

rq_size This is a number between 0x0 and 0xff for the count of entries in a device's job request queue.

cq_size This is a number between 0x0 and 0xff for the count of entries in a device's job completion queue.

boot device This field determines whether a device may be used to boot programs. A "1" means that it is bootable; a "0" means that it is not.

word size This field shows the word size of a device I/O bus. A "1" is used for devices with a 16 bit bus word; a "0" is used for devices with an 8 bit bus word.

brd size This field specifies the I/O connector slots that a device requires. A "1" indicates that two slots are needed; while a "0" means that one is required.

smart board This field determines whether a device is intelligent, i.e., requires downloaded code for normal operation or supports subdevices. A "1" indicates an intelligent device, while a "0" specifies a "dumb" device.

cons_cap This field shows whether a device can support the system console terminal. A "1" is used for devices that can; a "0" for those that cannot.

cons_file This field shows whether a device requires pump code to provide a system console interface. A "1" in this field means that the board cannot support the console interface without extra code. This field may have the "1" value only when the **cons_cap** field does also. A "0" in this field means that the device can support a system console terminal with PROM-based code when **cons_cap** has the value "1". This field must have a "0" value when **cons_cap** is "0".

Subdevices

ID code This is a number between 0x0 and 0xffff for the code that identifies a subdevice. Subdevice ID codes are administered by AT&T Technologies.

subdev name This field is a string (maximum of 9 characters) for a subdevice name. Subdevice names are uppercase and are administered by AT&T Technologies.

dev name This field is a string (maximum of 9 characters) for the device name to which a subdevice is associated. If a device table entry is to be removed, associated subdevice table entries may also be removed in a separate program call. The device name is necessary for an Equipped Device Table (EDT) check that will verify that a subdevice table entry is needed only for a device entry that is to be removed.

EXAMPLES

Generate and list the base entries for both the device and subdevice tables, saving the results in **./edt_data**.

```
edittbl -g -l -s -d
```

Install subdevice entries with new ID codes from the file **subdev.in** into the existing file **./edt_data**.

```
edittbl -i -s < subdev.in
```

List the device table entries found in an existing copy of the file that DGMON loads, the INIT file system **edt_data** file.

```
edittbl -l -d /init/dgn/edt_data
```

NAME

edt - Equipped Device Table commands

SYNOPSIS

edt

q

DESCRIPTION

Edt is a firmware command that displays the fields of the Equipped Device Table (EDT) entries, device by device. *Edt* is useful when a check of the exact contents of the EDT will resolve uncertainties.

q(uit) provides a return to the Maintenance Control Program (MCP) null mode (or the Debug Monitor (DEMON), if equipped).

SEE ALSO

filledt(8)

NAME

filledt – fill Equipped Device Table (EDT)

SYNOPSIS

filledt

DESCRIPTION

Filledt is a firmware-level routine that completes the Equipped Device Table (EDT) for the computer. It compares ID codes gathered for devices and subdevices to those in the look-up tables stored in */dgn/edt_data* and copies data, such as device names, into the EDT when ID matches between the EDT and the look-up table occur. *Filledt* also makes a console terminal check.

Filledt has two operating modes: power-up (automatic) and manual.

During the power-up sequence it silently completes the EDT and checks the console. An error message, **EDT FILL FAILED**, appears only if a peripheral device fails to respond to subdevice equipage queries.

In the manual request mode *filledt* completes the EDT, reporting success or failure. It checks for a console terminal and reports the console location and c_cflags values.

FILES

/filledt

SEE ALSO

termio(7).

NAME

newkey - makes a floppy key for the 3B2 Computer

SYNOPSIS

newkey

q

DESCRIPTION

Newkey is a firmware command that uses a formatted floppy disk to store the key that clears the Non-Volatile RAM (NVRAM) during the power-up or system reset sequences. The floppy disk on which the key resides is called the '*floppy key*'. It is often used to reset the firmware passwd to its default, *mcp*.

Q(uit) provides a return to the Maintenance Control Program (MCP) null mode (or the Debug Monitor (DEMON), if equipped).

EXAMPLE

Enter name of program to execute [ ]: **newkey**
Creating a floppy key to enable clearing of the saved NVRAM information
Insert a formatted floppy, then type 'go' (q to quit): **go**
Creation of floppy key complete

NAME

passwd - change firmware password

SYNOPSIS

passwd

q

DESCRIPTION

The *passwd* firmware command re-assigns the firmware password for a 3B2 computer. It is similar to the system-level command. A firmware password may not exceed 8 characters.

EXAMPLE

Enter name of program to execute [ ]: **passwd**
enter old passwd: <*old passwd*>
enter new passwd: <*new passwd*>
confirmation: <*new passwd*>

SEE ALSO

passwd(1) in the *AT&T 3B2 Computer User Reference Manual.*

NAME

ports – create character device files and inittab entries for ports boards automatically

SYNOPSIS

ports

DESCRIPTION

Ports will create character device files in **/dev** and add new entries in **/etc/inittab** for 4 asynchronous RS-232 ports and 1 parallel printer port. Each port will be named

tty[slot#][1-5]

If the board configuration has changed, *ports* will do the following:

Remove any tty device files for a board that no longer resides in a slot.

Remove device files of other boards such as the NI if a ports board now resides in the slot that previously held an NI. A warning message would be sent to the console that a device file was being removed.

Make new tty device files for the ports boards if needed.

Make new inittab entries for the ports boards.

If the configuration has not changed, the ports program will exit without doing anything.

Any devices, such as a printer or a modem, that are added to a ports board should link the names that are to be used for the devices to the corresponding tty device files that were created [see *ln*(1)].

EXAMPLE

A parallel printer is added to a ports board that is in slot 1. The corresponding slot would be tty15. The user should use *ln* to link an appropriate name such as lp1 to the tty device file.

ln /dev/tty15 /dev/lp1

FILES

/etc/inittab
/dev/tty[stol#][1-5]

SEE ALSO

cp(1) in the *AT&T 3B2 Computer User Reference Manual.*

WARNINGS

A warning will be issued for each device file that is removed provided it is not a tty device file. If a ports board has been removed and lp1 has been linked to a tty device file, the message would be as follows:

Warning: /dev/lp1 is being removed.

NAME

sysdump - save the memory image of the kernel after system panic

SYNOPSIS

sysdump

q

DESCRIPTION

The *sysdump* firmware command permits the user to capture an image of the 3B2 computer's main memory after a system panic. To use *sysdump* the user must already have enough formatted floppy disks, 700Kb each, to store the memory image.

The *sysdump* command dumps the system memory image to one or more floppy disks depending upon the size of memory and the user request. This memory image can later be analyzed by using *crash*(1M). *Sysdump* is invoked as a boot option.

When booted, *sysdump* begins an interactive procedure that prompts the user to insert the floppies to be loaded. The user has the option of quitting the session any time. This allows the user to dump only the portion of the system image that is needed.

The output of *sysdump* provides one input to *crash*(1M). The other input is the text file that was used to boot this system image. This is needed to provide symbolic reference to the system dump. The text file must be manually saved after the machine has been booted. If /unix was booted, then this should be dumped to floppy to accompany the system dump.

EXAMPLE

Enter name of program to execute [ ]: **sysdump**
Do you want to dump a system image to a floppy disk?
Enter 'c' to continue, 'q' to quit:**c**

Insert first sysdump floppy.
Enter 'c' to continue, 'q' to quit:**c**

Dumping mainstore.....

If you wish to dump more of mainstore insert new floppy

Enter 'c' to continue, 'q' to quit: **c**

Dumping more mainstore.....
Dumping completed.
2 floppies written

Returning to firmware

SELF-CHECK

FIRMWARE MODE

SEE ALSO

AT&T 3B2 Computer Crash Analysis Guide.

NAME

version - version commands

SYNOPSIS

version

q

DESCRIPTION

The *version* firmware command lists information about the ROM code in the 3B2 system board. It prints data about the ROM issue, date, software load and serial number.

Q(uit) provides a return to the Maintenance Control Program (MCP) null mode (or the Debug Monitor (DEMON), if equipped).