'F'. Thus, **$(@D)** refers to the directory part of the string **$@**. If there is no directory part, "./" is generated. The only macro excluded from this alternative form is **$?**. The reasons for this are debatable.

Suffixes

Certain names (for instance, those ending with '.o') have inferable prerequisites such as '.c', '.s', etc. If no update commands for such a file appear in *makefile*, and if an inferable prerequisite exists, that prerequisite is compiled to make the target. In this case, *Make* has *inference* rules which allow building files from other files by examining the suffixes and determining an appropriate *inference rule* to use. The current default inference rules are:

.c .c~ .sh .sh~ .c.o .c~.o .c~.c .s.o .s~.o .y.o .y~.o .l.o .l~.o .y.c .y~.c .l.c .c.a .c~.a .s~.a .h~.h

(The internal rules for make are contained in the source file "rules.c" for the make program. These rules are expected to be locally modified. To print out the rules compiled into the make on any machine in a form suitable for recompilation the following command is used:

make -fp - 2>/dev/null </dev/null

The only peculiarity in this output is the "(null)" string which printf(3) prints when handed a null string.)

The tilda in the above rules refers to an SCCS file. Thus, the rule ".c~.o" would transform an SCCS C source file into an object file (".o"). Since the "s." of the SCCS files is a prefix it is incompatible with the "suffix" point of view of make. Hence, the tilda is a way of changing any file reference into an SCCS file reference.

A rule with only one suffix (i.e. ".c:") is the definition of how to build "x" from "x.c". In effect, the other suffix is null. This is useful for building targets from only one source file. (e.g. shell procedures, simple C programs).

Additional suffixes are given as the dependency list for **.SUFFIXES**. Order is significant; the first possible name for which both a file and a rule exist is inferred as a prerequisite. The default list is:

.SUFFIXES: .o .c .y .l .s

(Here, again, the above command for printing the internal rules will display the list of suffixes implemented on the current machine.) Multiple suffix lists accumulate; **.SUFFIXES:** with no dependencies clears the list of suffixes.

Inference Rules

The first example can be done more briefly:

```
pgm: a.o b.o
        cc a.o b.o −o pgm
a.o b.o: incl.h
```

This is because make has a set of internal rules for building files. The user may add rules to this list by simply putting them in the *makefile*.

Certain macros are used by the default inference rules to permit the inclusion of optional matter in any resulting commands. For example, **CFLAGS**, **LFLAGS**, and **YFLAGS** are used for compiler options to *cc*, *lex* and *yacc*(1) respectively. Again, the previous method for examining the current rules is recommended.

The inference of prerequisites can be controlled. The rule to create a file with suffix *.o* from a file with suffix *.c* is specified as an entry with ".c.o:" as the 'target' and no dependents. Shell commands associated with the target define the rule for making a ".o" file from a ".c" file. Any target which has no slashes in it and starts with a dot is identified as a rule and not a true target.

Libraries

If a target or dependency name contains parenthesis, it is assumed to be an archive library, the string within parenthesis referring to a member within the library. Thus "lib(file.o)" and "$(LIB)(file.o)" both refer to an archive library which contains "file.o". (This assumes the "LIB" macro has been previously defined.) The expression "$(LIB)(file1.o file2.o)" is not legal. Rules pertaining to archive libraries have the form ".XX.a" where the "XX" is the suffix from which the archive member is to be made. An unfortunate byproduct of the current implementation requires the "XX" to be different from the suffix of the archive member. Thus, one cannot have "lib(file.o)" depend upon "file.o" explicitly. The most common use of the archive interface is as follows (Here, we assume the source files are all C type source):

```
lib:    lib(file1.o) lib(file2.o) lib(file3.o)
            @echo lib is now up to date
.c.a:
            $(CC) -c $(CFLAGS) $<
            ar rv $@ $*.o
            rm -f $*.o
```

In fact, the ".c.a" rule listed above is built into make and, thus, is unnecessary in this example. A more interesting, but more limited example of an archive library maintenance construction follows:

```
lib:    lib(file1.o) lib(file2.o) lib(file3.o)
            $(CC) -c $(CFLAGS) $(?:.o=.c)
            ar rv lib $?
            rm $?
            @echo lib is now up to date
.c.a:;
```

Here the substitution mode of the macro expansions is used. The **$?** list is defined to be the set of object file names (inside "lib") whose C source files are out of date. The substitution mode translates the ".o" to ".c". (Unfortunately, one cannot as yet transform to ".c~"; however this will come also.) Note also, the disabling of the ".c.a:" rule, which would have created each object file, one by one. This particular construct speeds up archive library maintenance considerably. This type of construct becomes very cumbersome if the archive library contains a mix of assembly programs and C programs.

Options and Other Stuff

The following is a brief description of all options and some special names:

-f file Description file name. The next argument is assumed to be the name of a description file. A file name of "-" denotes the standard input. The contents of the description files override the built-in rules if they are present.

-p Print out the complete set of macro definitions and target descriptions

-i Ignore error codes returned by invoked commands. This mode is entered if the fake target name ".IGNORE" appears in the description file.

-k abandon work on the current entry, but continue on other branches that do not depend on that entry.

-s Silent mode. Do not print command lines before executing. This mode is also entered if the fake target name ".SILENT" appears in the description file.

-r	Do not use the built-in rules.
-n	No execute mode. Print commands, but do not execute them. Even lines beginning with an "@" sign are printed.
-b	Compatability mode for old makefiles.
-e	Environment variables override assignments within makefiles.
-m	Print a memory map showing text, data, and stack. This option is a no-op on systems without the *getu(2)* system call.
-t	Touch the target files (causing them to be up to date) rather than issue the usual commands.
-d	Debug mode. Print out detailed information on files and times examined.
-q	Question. The **make** command returns a zero or non-zero status code depending on whether the target file is or is not up to date.
.DEFAULT:	If a file must be made but there are no explicit commands or relevant built-in rules, the commands associated with the name ".DEFAULT" are used if it exists.
.PRECIOUS:	Dependents of this target will not be removed when quit or interrupt are hit.
.SILENT:	Same effect as the -s option.
.IGNORE	Same effect as the -i option.

FILES

makefile, Makefile, s.makefile, s.Makefile

SEE ALSO

sh(1)

S. I. Feldman *Make − A Program for Maintaining Computer Programs,* Bell Laboratories Computing Science Tech. Rep. No. 57, April, 1977

E. G. Bradford − An Augmented Version of Make, TM 79-5255-1, July, 1979

BUGS

Some commands return nonzero status inappropriately. Use −i to overcome the difficulty.
Commands that are directly executed by the Shell, notably *cd*(1), are ineffectual across newlines in *make*.
The syntax "(lib(file1.o file2.o file3.o)" is illegal.
lib(file.o) cannot be built from file.o.
The macro $(a:.o=.c~) doesn't work.

NAME

man − print pages of this manual

SYNOPSIS

man [section] [−options] [**all**] [title ...]

DESCRIPTION

Man is a shell command file that will locate and run off individual pages of this manual. The meaning of the parameters are:

section Sections of the manual to be searched for *title*(s). If no *section*(s) are specified *man* assumes that it was called with:

man 1 2 3 4 5 6 7 8 title ...

If no sections are specified, *man* looks for *titles* in the current directory before looking at any regular manual sections. *Section*(s) are searched in the order given and only those sections given are searched. All *sections* to be searched must be specified before any *titles* are given.

−options The following *options* have special meaning to *man*:

t causes output to be prepared for use with the phototypesetter simulator *tc*(1) using *troff*(1).

g causes output to be prepared for use with the *gcat*(1) program which produces output from the phototypesetter.

T43 causes output to be prepared for printing on the TTY43 teleprinter. The width is set to 51 and the length is set to 100.

TVP causes output to be prepared for printing on the Versatec printer. The width is set to 100.

[Ll] causes *man* to search only the local section(s) **/usr/man/local/man?**.

[Cc] causes man to search only the general section(s) **/usr/man/cbunix/man?**.

All other options given to *man* are passed to *nroff*. The default options are set up to be as general as possible, thus *man* will work for most devices without the need for any options. Also, *man* makes use of the post-processing program, *over*, to prepare its output for the most general case (except that *over* is disabled for CRTs); if any *options* are specified then *over* is not used.

all *All* instructs *man* to print all the manual pages in the specified sections. However, if no *section*(s) are supplied, *all* is an error. This forces the user to specifically request sections to be printed under the *all* option.

title Names of the manual page(s) to be printed. Generally, the title of a 'thing' is similar to the name which must be used to access the 'thing'. For instance:

man man

will reproduce this page.

General Information:

There are several options with special meaning. The **−t** option causes output to be prepared for use with the phototypesetter simulator *tc*(1). The **−g** options causes output to be prepared for use with the *gcat*(1) program which produces output from the phototypesetter. Finnally, the **−T43** option causes output to be prepared for printing on a tty43 teleprinter.

The manual usually resides on a mountable file system which may not always be mounted. If it is not mounted the diagnostic message:

Manual pack not mounted

will be printed on the error output device (usually the terminal from which *man* is run).

The manual is organized into 8 sections; some of which have various sub-sections. Each section has one section which is reserved for use by local groups. The current sections are:

1	UNIX Commands.
2	System Calls.
3	Subroutines.
4	Device Interfaces and Special Files.
5	File Formats, Tables and Macros.
6	UNIX System Explanations.
7	Kinks and Conventions.
8	Stand-alone Utilities.

The *man* command may be used to print manual pages in your own directory. If a manual page:

junk.3

exists in the present working directory (see *pwd*(1)) and a *man* command of the form:

man junk.3

will print the *junk* manual page — without looking through the 'normal' manual sections.

If there is both a 'local' and a 'basic' version of a manual page then *man* will print the 'local' version unless the search order built into *man* is overridden by the user.

FILES

.	present working directory
/usr/man/local/*section*/*	'local' versions of manual pages
/usr/man/cbunix/*section*/*	'basic' versions of manual pages

SEE ALSO

over(1), col(1), nroff(1), manmac(5)

BUGS

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

NAME

mesg − permit or deny messages

SYNOPSIS

mesg [n] [y]

DESCRIPTION

Mesg with argument **n** forbids messages via *write*(1) by revoking non-user write permission on the user's terminal. *Mesg* with argument **y** reinstates permission. All by itself, *mesg* reports the current state without changing it.

FILES

/dev/ln*

SEE ALSO

write(1)

DIAGNOSTICS

Exit status is 0 if messages are receivable, 1 if not, 2 on error.

NAME

mhdump − incremental file system dump

SYNOPSIS

mhdump [key [arguments] filesystem]

DESCRIPTION

Mhdump makes an incremental file system dump on magtape of all files changed after a certain date. The *key* argument specifies the date and other options about the dump. *Key* consists of characters from the set **abcfiu0hdsn**.

- **a** Normally files larger than 500 blocks are not incrementally dumped; this flag forces them to be dumped.
- **b** The next argument is taken to be the density of the dump tape (i.e., 800 or 1600).
- **c** If the tape overflows, increment the last character of its name and continue on that drive. (Normally it asks you to change tapes.)
- **f** Place the dump on the next argument file instead of the tape.
- **i** the dump date is taken from the entry in the file **/etc/dtab** corresponding to the last time this file system was dumped with the **−u** option.
- **u** the date just prior to this dump is written on **/etc/dtab** upon successful completion of this dump. This file contains a date for every file system dumped with this option.
- **0** the dump date is taken as the epoch (beginning of time). Thus this option causes an entire file system dump to be taken.
- **h** the dump date is some number of hours before the current date. The number of hours is taken from the next argument in *arguments*.
- **d** the dump date is some number of days before the current date. The number of days is taken from the next argument in *arguments*.
- **s** the size of the dump tape is specified in feet. The number of feet is taken from the next argument in *arguments*. When the specified size is reached, the dump will wait for reels to be changed. The default size is 2200 feet.
- **n** Normally, a name list generated by *ncheck*(1M) is placed on the tape so that *mhrestor*(1M) may extract files by name. This flag suppresses the generation of names.

If no arguments are given, the *key* is assumed to be **i** and the file system is assumed to be **/dev/rp0**.

Full dumps should be taken on quiet file systems as follows:

```
mhdump 0u /dev/rp0
```

Incremental dumps should then be taken when desired by:

```
mhdump
```

When the incremental dumps get cumbersome, a new complete dump should be taken. In this way, a restore requires loading of the complete dump tape and only the latest incremental tape.

DIAGNOSTICS

If the dump requires more than one tape, it will ask you to change tapes. Reply with a newline when this has been done. If the first block on the new tape is not writable, e.g., because you forgot the write ring, you get a chance to fix it. Generally, however, read or write failures are fatal.

FILES

/dev/rmt0:	magtape
/dev/rp0:	default file system
/etc/dtab:	record of last full dump
/etc/ncheck	

SEE ALSO

mhrestor(1M), ncheck(1M), dump(5)

BUGS

It's slow.

It does not work for file systems larger than 64K blocks.

NAME

mhrestor – incremental file system restore

SYNOPSIS

mhrestor key [argument ...]

DESCRIPTION

Mhrestor is used to read magtapes dumped with the *mhdump* command. The *key* specifies what is to be done. *Key* is one of the characters **rxt** optionally combined with **f**.

f Use the first *argument* as the name of the tape instead of the default.

r The tape is read and loaded into the file system specified in *argument*. This should not be done lightly (see below).

x Each file on the tape named by an *argument* is extracted. The file extracted is placed in a file with a numeric name supplied by *mhrestor* (actually the inode number). In order to keep the amount of tape read to a minimum, *mhrestor* asks you to 'mount the desired tape volume' and tell *mhrestor* which volume you put on the drive. On a multivolume dump the recommended procedure is to mount the last through the first volume in that order. *Mhrestor* checks to see if any of the files requested are on the mounted tape (or a later tape, thus the reverse order) and doesn't read through the tape if no files are. If you are working with a single volume dump or the number of files being restored is large, respond to the query with '1' and *mhrestor* will read the tapes in sequential order.

t Print the date the tape was written and the date the filesystem was dumped from.

The **r** option should only be used to restore a complete dump tape onto a clear file system or to restore an incremental dump tape onto this. Thus

```
/etc/mkfs /dev/rp0 40600
mhrestor r /dev/rp0
```

is a typical sequence to restore a complete dump. Another *mhrestor* can be done to get an incremental dump in on top of this.

A *mhdump* followed by a *mkfs* and a *mhrestor* is used to change the size of a file system.

FILES

default tape unit varies with installation

SEE ALSO

mhdump(1M), mkfs(1M), dump(5)

DIAGNOSTICS

There are various diagnostics involved with reading the tape and writing the disk. There are also diagnostics if the i-list or the free list of the file system is not large enough to hold the dump.

If the dump extends over more than one tape, it may ask you to change tapes. Reply with a new-line when the next tape has been mounted.

BUGS

There is redundant information on the tape that could be used in case of tape reading problems. Unfortunately, *mhrestor* doesn't use it.

NAME

mhstty − set the options for a terminal

SYNOPSIS

mhstty [−**g**] [option ...]

DESCRIPTION

Mhstty sets certain I/O options on the current input terminal. With no argument, it reports the current settings of the options. With the −**g** flag, it reports current settings in a format that can be used as arguments to another *mhstty* command. The option strings are selected from the following set:

even allow even parity.
−even disallow even parity.
odd allow odd parity.
−odd disallow odd parity.
raw raw mode input (no erase, kill, interrupt, quit, EOT; parity bit passed back).
−raw negate raw mode.
cooked same as **−raw**.
nl accept only new-line to end input lines.
−nl allow carriage return for new-line, and output CR-LF for carriage return or new-line.
echo echo back every character typed.
−echo do not echo characters.
lcase or **LCASE** map upper case to lower case.
−lcase or **−LCASE** do not map upper case to lower case.
tabs preserve tabs when printing.
−tabs replace tabs by spaces when printing.
erase *c* set erase character to *c*.
kill *c* set kill character to *c*.
ek reset erase and kill characters back to normal # and @.
cr0 cr1 cr2 cr3 select style of delay for carriage return (see *ioctl*(2)).
nl0 nl1 nl2 nl3 select style of delay for line-feed (see *ioctl*(2)).
tab0 tab1 select style of delay for tab (see *ioctl*(2)).
bs0 bs1 select style of backspace delay.
lfkc echo an LF after a kill character.
−lfkc do not echo an LF after a kill character.
ff0 ff1 select style of delay for form-feed (see *ioctl*(2)).
tty33 set all modes suitable for the TELETYPE® Model 33 terminal.
tty37 set all modes suitable for the TELETYPE Model 37 terminal.
vt05 set all modes suitable for Digital Equipment Corp. VT05 terminal.
tn300 set all modes suitable for a General Electric TermiNet 300.
ti700 set all modes suitable for Texas Instruments 700 series terminal.
tek set all modes suitable for Tektronix 4014 terminal.
hup hang up DATA-PHONE® connection on last close.
−hup do not hang up DATA-PHONE connection on last close.
0 hang up phone line immediately.
50 75 110 134 150 200 300 600 1200 1800 2400 4800 9600 exta extb Set terminal baud rate to the number given, if possible. (These are the speeds supported by the DH-11 interface).

SEE ALSO

tabs(1), ioctl(2).

NAME

mkconf − create configuration table and low core

SYNOPSIS

mkconf [**−nta**] [address]

DESCRIPTION

Mkconf creates two files, one which contains the low core vectors (**l.s**) and one which contains the configuration tables (**c.c**), needed in creating a UNIX. It is an interactive program which creates these two files from the input specifications supplied by the user. The specifications allow the input of the type of device, the number of a given type and the vector address if the device is in the floating vector area. The root and swap devices may be specified along with the minor device number of the devices, and the location and size of the swap area (swplo and nswap) values may also be specified. The first input to *mkconf* must be the type of processor. The following is an example of how *mkconf* is used. Notice that the devices specified as the root and swap must already be configured.

```
mkconf
70
hp
hs
ht
2dc
root hp 1
swap hs 8
swplo 1
nswap 4000
```

Typing *list* before hitting CTL D will produce a list of vector addresses and the devices at those locations. Typing an EOT (CTL D) terminates input and causes **l.s** and **c.c** to be generated.

Mkconf automatically adds a jump to a routine whenever there is an empty location in low core. This routine prints the message

stray interrupt at XXX

when an interrupt occurs at one of these locations. An interrupt occurring at any of these locations is treated similarly to a device interrupt. The stray interrupt routine will print the location (modulo 128) of the interrupt, thus making it possible to narrow the number of locations down. The arguments to *mkconf* alter the use of the jump to the stray interrupt routine in the following ways.

−n No stray interrupt vectors are produced.

−t A jump to trap+15 is used instead of stray. This is not quite so useful but it is better than nothing.

−a Currently, the stray interrupts are produced up to location 0400. The second argument (address) can extend or shorten this area.

NAME

mkdir – make a directory

SYNOPSIS

mkdir dirname ...

DESCRIPTION

Mkdir creates specified directories in mode 777, possibly modified by the current *umask*(1). Standard entries, '.', for the directory itself, and '..', for its parent, are made automatically.

Mkdir requires write permission in the parent directory.

SEE ALSO

rm(1), umask(1)

DIAGNOSTICS

Mkdir returns exit code 0 if all directories were successfully made. Otherwise it prints a diagnostic and returns nonzero.

NAME

mkfs − construct a file system

SYNOPSIS

/etc/mkfs special proto [**−b** [numbers]]

DESCRIPTION

Mkfs constructs a file system by writing on the special file *special* according to the directions found in the prototype file *proto*. 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 (see *bproc*(6)). The second token is a number specifying the size of the created file system. Typically it will be the number of blocks on the device, perhaps diminished by space for swapping. The next token is the i-list size in blocks (remember there are 16 i-nodes per block). An optional third token is the keyword **badblocks** followed by a list of numbers(decimal) and terminated with the token $. The list specifies blocks that are to be left out of the file system. 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 three digit octal number giving the owner, group, and foreigner read, write, execute permissions (see *chmod*(1)).

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

If the file is a regular file, the next token is a pathname 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) file specifications for the entries in the directory. The scan is terminated with the token **$**.

If the prototype file cannot be opened and its name consists of 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 *proto* interpreted as a decimal number. The i-list size is the file system size divided by 50. (This corresponds to an average size of three blocks per file.) The boot program is left uninitialized.

The **−b** option allows specification of bad block numbers on the command line. This may be used in combination with the **badblocks** keyword in the prototype file.

A sample prototype specification follows:

```
/usr/mdec/uboot
4872 55
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
        $
$
```

or for a file system with 3 bad blocks,

```
/usr/mdec/util/uboot
4000 55
badblocks 67 106 2004 $
d--777 3 1
$
```

SEE ALSO

mkpt(1M), dir(5), fs(5), bproc(6)

DIAGNOSTICS

There are various diagnostics for syntax errors, inconsistent values, and sizes too small. Also, badblocks specified in the ilist or outside the file system are not accepted.

BUGS

It is not possible to initialize a file larger than 256K bytes.
There should be some way to specify links.
Check does not yet know about bad blocks. They are reported as missing and a salvage operation will replace them in the file system.

NAME

mkfst − construct a file system on mag tape

SYNOPSIS

/etc/mkfst special proto [recsiz]

DESCRIPTION

Mkfst constructs a file system by writing on the special file *special* according to the directions in the prototype file *proto*. See the description of *mkfs*(1M) for the details on building a proto file.

In fact, this program works exactly like *mkfs* with the following two exceptions:

1) *Mkfst* builds the inodes for the tape file system in a disk file before copying them to tape. Since, for large file systems, this temporary file would get very large, it is not recommended that *mkfst* be used as a general replacement for *mkfs*. It is anticipated that tape file systems will be fairly small.

2) *Mkfst* allows the user to specify the size of the records to be written on the tape. The optional argument **recsiz** specifies the number of disk blocks per tape record (a disk block is 256 words). If *recsiz* is not "1", the *special* file must be capable of physical I/O, for example **/dev/rmt***?*. This feature is useful for generating file systems which are meant to be eventually copied onto a mass storage device (eg RP03). The stand-alone utility used to dump and load the device to and from tape may require that tape records be larger than one disk block.

SEE ALSO

mkfs(1M)

FILES

/tmp/mtmp

DIAGNOSTICS

See *mkfs*.

NAME

mklfs − construct a Logical File System (LFS)

SYNOPSIS

/etc/mklfs [options] device nlfn ncyl trkf secf blkf

DESCRIPTION

Mklfs constructs a Logical File System on disk. The LFS software is implemented as a pseudo device driver which provides a simple file system implemented through the UNIX raw I/O facility. The LFS consists of a set of contiguously allocated files, specified by their logical file numbers, which are manipulated by LFS operations.

Options:
An optional character string consisting of a dash (−) followed by any combination of the characters **y**, **n**, or **v** whose meanings are as follows:

- **y** unconditionally make the LFS (no user interaction)
- **n** don't make the LFS (no user interaction)
- **v** verbose output (LFS header information).

device The block-special file corresponding to the physical disk area upon which the LFS will be built. Before executing *mklfs*, *device* must be created by the *mknod*(1) command.

nlfn The maximum number of logical file numbers (lfns) to be allowed. Legal lfns start at 1 and range up to and including *nlfn*. There is currently a limitation of 65,535 lfns per file system.

ncyl The number of cylinders in the LFS disk area.

trkf The number of tracks per cylinder.

secf The number of 512 byte sectors per track.

blkf The size of a LFS block; i.e., all files will be allocated in multiples of blkf sectors.

An example of *mklfs* follows:

/etc/mklfs /dev/lfs 40000 156 19 22 2

creates an LFS with 40,000 files spanning 156 cylinders for a DEC RPO4/5/6 disk (under CB-UNIX, 156 cylinders is the maximum size file system possible).

SEE ALSO

lfs(3C)

DIAGNOSICS

Diagnostics are given if *device* cannot be opened or is not a block device, if the wrong number of arguments are given, or if parameters are unintelligible.

LIMITATIONS

At most 65,535 logical files are allowed per file system. Each logical file system must be less than 32,768 LFS blocks in size. This is since the system *malloc* routine is used to manage free space, and the size is stored in a short integer. Making *blkf* larger can help compensate for this limitation.

NAME

mknod – build special file

SYNOPSIS

/etc/mknod name [**c**] [**b**] major minor

DESCRIPTION

Mknod makes a directory entry and corresponding i-node for a special file. The first argument is the *name* of the entry. 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).

The assignment of major device numbers is specific to each system. For reference, here are the numbers for the CB Operating System Group machine. Do not believe them too much.

Block devices:

0 RX01 Floppy Disk

1 Not used. Reserved for RP03 disk.

2 Not used. Reserved for RF disk.

3 Not used. Reserved for TM11 tape.

4 Not used. Reserved for DECtape.

5 RP06 Disk.

6 TE16 Tape.

Character devices:

0 System Console.

1 DZ11 Asynchronous Interface

2 Not used. Reserved for line printer.

3 Not used. Reserved for DC11 asynchronous interface.

4 DH11 Asynchronous Interface.

5 Not used. Reserved for DP interface.

6 Not used. Reserved for DJ11 asynchronous interface.

7 DN11 auto dial interf....ace.

8 Memory Access.

9 Not used. Reserved for raw RK disk interface.

10 Raw RX disk interface.

11 Not used. Reserved for raw RP03 disk interface.

12 Not used. Reserved for raw TM11 tape interface.

13 Raw RP06 disk interface.

14 Raw TE16 tape interface.

15 Not used. Reserved for HS interface.

16 Controlling TTY Interface.

17 Not used.

18 Named pipe Interface.

19 Not used.

20 Error log interface

SEE ALSO

mknod(2)

NAME

mkpt - make proto

SYNOPSIS

mkpt primer name

DESCRIPTION

mkpt produces a proto file in the output file, "name", based on a starting or primer file, "primer". The resulting proto in "name" can be used with the *mkfs*(1M) command. The primer consists of ACSII tokens separated by new line characters. All the tokens except the last one are copied as is to the output file. They have no meaning to the *mkpt* command. See the *mkfs*(1M) command for a more detailed description of the first six tokens. The tokens and their meanings are:

<bootfile>	The name of the boot file program.
<file size>	The number of blocks of the proto.
<i-list size>	The size of the i-list in blocks.
<mode>	The mode of the root directory of the proto.
<user id>	The user id of the root directory of the proto.
<group id>	The group id of the root directory of the proto.
<starting directories>	One or more directories from which the proto will be made. These are the only strings that have any meaning to mkpt. All the other strings are copied as is to the output file.

The mkpt command will ignore directory names beginning with a ".". A typical primer file is:

```
/usr/sys/mboot
4872
55
d--755
0
1
/bin
/usr/bin
```

SEE ALSO

mkfs(1M)

NAME

mm − type out documents that use the PWB/MM macros

SYNOPSIS

mm [options] [files]

DESCRIPTION

Mm can be used to type out documents using *nroff*(1) and the PWB/MM text formatting codes. It has options to specify preprocessing by *tbl*(1) and/or *neqn*(1) and postprocessing by various terminal-oriented output filters. The proper pipelines and the required arguments and flags for *nroff*(1) and PWB/MM are generated, depending on the options selected.

Options for *mm* are given below. Any other arguments or flags (e.g., **−rC3**) are passed to *nroff*(1) or to PWB/MM, as appropriate. Such options can occur in any order, but they must appear before the *files* arguments. If no arguments are given, *mm* prints a list of its options.

−T*term* Specifies the type of output terminal; recognized values for *term* are (see *term*(7)): **300**, **300s**, **450**, **300-12**, **300s-12**, **450-12**, **37**, **4014**, **hp**, **1520**, **745**, **43**, **tn300**, and **lp**. If this option is *not* used, *mm* will use the value of the shell variable **$TERM** from the environment (see *profile*(5) and *environ*(7)) as the value of *term*, if **$TERM** is set; otherwise, *mm* will use **300** as the value of *term*. If several terminal types are specified, the last one takes precedence.

−12 Indicates that the document is to be produced in 12-pitch. May be used when **$TERM** is set to one of **300**, **300s**, and **450**. (The pitch switch on the DASI 300 and 300s terminals must be manually switched to **12** if this option is used.)

−c Causes *mm* to invoke *col*(1); note that if *term* is one of **hp**, **1520**, **745**, **43**, **tn300**, and **lp**, then *col*(1) is automatically invoked by *mm*.

−e Causes *mm* to invoke *neqn*(1).

−t Causes *mm* to invoke *tbl*(1).

−E Invokes the **−e** option of *nroff*(1).

−u Causes *mm* to use the not pre-compiled version of the macros (see *mm*(7)).

As an example (assuming that the shell variable **$TERM** is set in the environment to **450**), the two command lines below are equivalent:

```
mm −t −rC3 −12 ghh*
tbl ghh* | nroff −cm −T450−12 −h −rW80 −r03 −rC3
```

Mm reads the standard input when **−** is specified instead of any file names. (Mentioning other files together with **−** leads to disaster.) This option allows *mm* to be used as a filter, e.g., "cat dws | mm −".

HINTS

1. *mm* usually invokes *nroff*(1) with the **−h** flag. With this flag, *nroff*(1) assumes that the terminal has tabs set every 8 character positions.
2. Use the **−o***list* option of *nroff*(1) to specify ranges of pages to be output.
3. If you use the **−s** option of *nroff*(1) (to stop between pages of output), use line-feed (rather than return or new-line) to restart the output.
4. If you lie to *mm* about the kind of terminal its output will (finally) be printed on, you'll get what you deserve: more or less subtle garbage.

SEE ALSO

col(1), env(1), eqn(1), nroff(1), tbl(1), profile(5), environ(7), mm(7), term(7).
PWB/MM − Programmer's Workbench Memorandum Macros by D. W. Smith and J. R. Mashey.
Typing Documents with PWB/MM by D. W. Smith and E. M. Piskorik.

DIAGNOSTICS

"mm: no input file" if none of the arguments is a readable file and *mm* is not used as a filter.

NAME

mmkdir – made path names

SYNOPSIS

mmkdir relative_path_name relative_path_name ...

DESCRIPTION

The *mmkdir* command will make all the directories in a relative path name input argument. If the directories already exist, then *mmkdir* does nothing. When successful, the relative path name for each input argument is printed on the standard output.

SEE ALSO

mkdir(1)

DIAGNOSTICS

All diagnostics are printed on file descriptor 2, and are hopefully self explanatory.

NAME

mo, mo90, nmo, nmo90 − nroff, nnroff mm interface for preprinted letterhead

SYNOPSIS

[n]mo[90] file [[F|D|T[PS]] [LP[F|D|T]]] [−x] [−[et]] [−s] [nroff options]

DESCRIPTION

Mo and *nmo*, which are based on commands developed on the CB PWB systems, are used for document preparation in Division 594. *Mo90* and *nmo90* use a page length of 90 lines. *Mo* eventually invokes *nroff*(1) with the −*mm* macro package; *nmo* invokes *nnroff* with the −*cm* compacted memorandum macro package. (*Nnroff* is based on UNIX 3.0 *nroff* and can use the newer macro packages, including the souped-up memorandum macros (*mm*), in their compacted form.) *Mo* is designed for use with pre-printed letterhead and thus always invokes *nroff* with −*rA1* −*rN2*. Correct format for these forms is insured by inclusion of the special file **/usr/lib/macros/mo.h**, containing *mm* macro calls, when *nroff* is called; if a file named **mo.h** exists in the current directory, it will be used instead. *Mo* also expects the **$TERM** shell variable to be set to the current terminal type for the −T*term nroff* argument.

Multiple *files* can be specified by including the list in double quotes, e.g., **intro part1 part2** . The capital letter keyletters can be interpreted as follows:

F The copy is intended to be a final copy, not a draft.

D|T The copy is a temporary or draft copy. (The existence of two flags for the same output is a historical accident.) *Nroff* will be invoked with the −*rC3* argument.

PS The output is destined for a DTC-382 terminal with a proportional spacing print wheel. *Nroff* is invoked with the −*e* argument and terminal type *ps96*.

LP The output is to be sent to *col*(1) and *over*(1) and the line printer spooler *lpr*(1).

The other *mo* arguments have the following interpretations:

−x The file will be sent through the *crypt*(1) program before being sent to *nroff*. The user will be asked to correctly enter the encryption key.

−e|t The file will be preprocessed by the *neqn*(1) *and/or tbl*(1) *nroff* preprocessors.

−s The −*s1* argument will be sent to *nroff* to cause it to pause after every complete output page and wait for a line feed indicating that the next output form has been correctly positioned.

nroff options

Any trailing *nroff* options will be passed along to the *nroff* program. None of the options mentioned above can be overridden by trying to change them with *options*. *Options* should seldom be necessary.

FILES

/usr/lib/macros/mo.h

SEE ALSO

col(1), crypt(1), lpr(1), neqn(1), nroff(1), over(1), tbl(1)

NAME

more, page - file perusal filter for crt viewing

SYNOPSIS

more [**-d**] [**-f**] [**-l**] [**-n**] [**+**linenumber] [**+/**pattern] [name ...]

page [**-d**] [**-f**] [**-l**] [**-n**] [**+**linenumber] [**+/**pattern] [name ...]

DESCRIPTION

More is a filter which allows examination of a continuous text one screenful at a time on a soft-copy terminal. It normally pauses after each screenful, printing --More-- at the bottom of the screen. If the user then types a carriage return, one more line is displayed. If the user hits a space, another screenful is displayed. Other possibilites are enumerated later.

The command line options are:

-n An integer which is the size (in lines) of the window which more will use instead of the default.

-d More will prompt the user with the message "Hit space to continue, Rubout to abort" at the end of each screenful. This is useful if more is being used as a filter in some setting, such as a class, where many users may be unsophisticated.

-f This causes more to count logical, rather than screen lines. That is, long lines are not folded. This option is recommended if nroff output is being piped through ul, since the latter may generate escape sequences. These escape sequences contain characters which would ordinarily occupy screen postions, but which do not print when they are sent to the terminal as part of an escape sequence. Thus more may think that lines are longer than they actually are, and fold lines erroneously.

-l Do not treat ^L (form feed) specially. If this option is not given, more will pause after any line that contains a ^L, as if the end of a screenful had been reached. Also, if a file begins with a form feed, the screen will be cleared before the file is printed.

+linenumber
Start up at linenumber.

+/pattern
Start up two lines before the line containing the regular expression pattern.

If the program is invoked as page, then the screen is cleared before each screenful is printed (but only if a full screenful is being printed), and k - 1 rather than k - 2 lines are printed in each screenful, where k is the number of lines the terminal can display.

More looks in the file /etc/termcap to determine terminal characteristics, and to determine the default window size. On a terminal capable of displaying 24 lines, the default window size is 22 lines.

If more is reading from a file, rather than a pipe, then a percentage is displayed along with the --More-- prompt. This gives the fraction of the file (in characters, not lines) that has been read so far.

Other sequences which may be typed when more pauses, and their effects, are as follows (i is an optional integer argument, defaulting to 1) :

i<space>
display i more lines, (or another screenful if no argument is given)

^D display 11 more lines (a ``scroll''). If i is given, then the scroll size is set to i.

d same as ^D (control-D)

iz same as typing a space except that i, if present, becomes the new window size.

is skip i lines and print a screenful of lines

if skip i screenfuls and print a screenful of lines

q or Q
Exit from more.

= Display the current line number.

v Start up the editor vi at the current line.

h Help command; give a description of all the more commands.

i/expr
search for the i-th occurrence of the regular expression expr. If there are less than i occurrences of expr, and the input is a file (rather than a pipe), then the position in the file remains unchanged. Otherwise, a screenful is displayed, starting two lines before the place where the expression was found. The user's erase and kill characters

may be used to edit the regular expression. Erasing back past the first column cancels the search command.

in search for the i-th occurrence of the last regular expression entered.

' (single quote) Go to the point from which the last search started. If no search has been performed in the current file, this command goes back to the beginning of the file.

!command
invoke a shell with command. The characters '%' and '!' in "command" are replaced with the current file name and the previous shell command respectively. If there is no current file name, '%' is not expanded. The sequences "\%" and "\!" are replaced by "%" and "!" respectively.

i:n skip to the i-th next file given in the command line (skips to last file if n doesn't make sense)

i:p skip to the i-th previous file given in the command line. If this command is given in the middle of printing out a file, then more goes back to the beginning of the file. If i doesn't make sense, more skips back to the first file. If more is not reading from a file, the bell is rung and nothing else happens.

:f display the current file name and line number.

:q or :Q
exit from more (same as q or Q).

. (dot) repeat the previous command.

The commands take effect immediately, i.e., it is not necessary to type a carriage return. Up to the time when the command character itself is given, the user may hit the line kill character to cancel the numerical argument being formed. In addition, the user may hit the erase character to redisplay the --More--(xx%) message.

At any time when output is being sent to the terminal, the user can hit the quit key (normally control-\). More will stop sending output, and will display the usual --More-- prompt. The user may then enter one of the above commands in the normal manner. Unfortunately, some output is lost when this is done, due to the fact that any characters waiting in the terminal's output queue are flushed when the quit signal occurs.

The terminal is set to noecho mode by this program so that the output can be continuous. What you type will thus not show on your terminal, except for the / and ! commands.

If the standard output is not a teletype, then more acts just like cat, except that a header is printed before each file (if there is more than one).

A sample usage of more in previewing nroff output would be

```
nroff -ms +2 doc.n | more
```

AUTHOR

Eric Shienbrood

FILES

/etc/termcap	Terminal data base
/usr/lib/more.help	Help file

SEE ALSO

script(1)

NAME

moo – guessing game

SYNOPSIS

/usr/games/moo

DESCRIPTION

Moo is a guessing game imported from England. The computer picks a number consisting of four distinct decimal digits. The player guesses four distinct digits being scored on each guess. A 'cow' is a correct digit in an incorrect position. A 'bull' is a correct digit in a correct position. The game continues until the player guesses the number (a score of four bulls).

BUGS

Watch out for the random number generator.

NAME

mount − mount file system

SYNOPSIS

mount special file **[ro] [restricted]**

DESCRIPTION

Mount announces to the system that a removable file system is present on the device corresponding to special file *special* (which must refer to a disk or possibly DEC tape). The *file* must exist already; it becomes the name of the root of the newly mounted file system.

Mount maintains a table of mounted devices; if invoked without an argument it prints the table.

The optional argument **ro** indicates that the file 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. The *mount* command is reserved by the operating system to the super-user. However, if the set-user-ID bit is turned on, the command has global use. If the third argument is not given, then the device indicated by *special* is checked to see if the writer permission mode is on for the owner, the owner's group, or for anyone. If no one had write access, then the file system is mounted as read-only and a message to this effect is printed to remind the user he had mounted a read-only file system.

The **restricted** argument indicates that the file system is to be marked so that the "set user/group id" feature of the *exec* system call is disabled. Programs marked to set the user or group id upon execution, if found on a file system so marked, will execute, but the setuid or setgid will not take place. Furthermore on a **restricted** file system, it is not possible to open character or block special devices.

All file systems mounted by users who are not *root* and who are not a part of the privileged systems groups, currently group *sys* and *superg*, will automatically have the **restricted** flag turned on if they mount a file system.

Furthermore, the *mount* command will only mount files systems for the unprivileged on the directories listed in the file **/etc/mountpts**. Only the privileged users may mount on any directory and without the **restricted** feature being specified.

FILES

/etc/mtab,
/etc/mountpts

SEE ALSO

umount(1), mount(2), mountpts(5), mtab(5)

NAME

move - move a file and set the mode

SYNOPSIS

move source mode uid gid [dest1 dest2 ...]

DESCRIPTION

The *move* command will move the given source file to the first destination file and link the destination file to all other destination files. The *move* command will also set the mode, user id, and group id (mode, uid, and gid, respectively) of the destination. If there are no destination files specified or if one of the destination files is the same as the source file *move* will leave the original file intact but change the mode, user id, and group id. Otherwise the source file is removed. The *cpmv* command exists to do the same function as move without removing the source file.

The mode is a three to six character string of the form *abcdef* where:

- *a* is a **u** (for set user id), - (for no-op), or missing (same as -)
- *b* is a **g** (for set group id), - (for no-op) or missing (same as -).
- *cde* is a three digit octal string specifing the read/write/execute permissions for self, group, and others.
- *f* is a **s** (for save text) or missing.

For example:

644 or **u-705** or --775s.

The user id and group id may be ASCII strings or numerical id's. Unlike the *mv* command, the destination arguments may not be directories.

FILES

/bin/ls

SEE ALSO

cpmv(1)

DIAGNOSTICS

All diagnostics are printed on file descriptor 2.

NAME

mtm − magnetic tape manipulation

SYNOPSIS

mtm [−sn] [−lm] [−bp] [unit]

DESCRIPTION

Mtm helps in the processing of multifile magnetic tapes. The optional arguments are:

−s*n* Forward space the magnetic tape for *n* files.

−l*m* Produce a list of the number and sizes of the records on the magnetic tape for *m* files. If *m* is missing *mtm* will analyze records up to a double EOT.

−b*p* Define the maximum record size as *p*K bytes. If *b* is missing, then the maximum record size is assumed to be 2K bytes.

unit *Unit* specifies the drive on which the tape is mounted. If *unit* is missing, drive 0 is assumed.

As an example,

mtm −s1 −12 1

gives as output:

File 2:
Record 1 − 14 bytes
Record 2 − 512 bytes
23 Records
File 3:
Record 1 − 512 bytes
68 Records
DONE

This means that file 2 contains one 14−byte record and twenty-two 512−byte records.

The two arguments can be combined to skip some files, then list some number of remaining files. If neither argument is given, *mtm* will list the entire tape.

FILES

/dev/rmt?

Mtm assumes the names of the tape unit special files that match their minor device numbers; thus it will try to add 4 to the specified *unit* number to obtain the corresponding device with no rewind.

NAME

mvdir – move a directory

SYNOPSIS

/etc/mvdir dirname name

DESCRIPTION

Mvdir renames 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).

Only super-user can use *mvdir*.

SEE ALSO

mkdir(1)

NAME

nar — new format archive and library maintainer

SYNOPSIS

nar key [posname] afile name ...

DESCRIPTION

Nar maintains groups of files combined into a single archive file. Its main use is to create and update library files as used by the loader. It can be used, though, for any similar purpose.

Key is one character from the set **drqtpmx,** optionally concatenated with one or more of **vuaibcl.** *Afile* is the archive file. The *names* are constituent files in the archive file. The meanings of the *key* characters are:

d Delete the named files from the archive file.

r Replace the named files in the archive file. If the optional character **u** is used with **r,** then only those files with modified dates later than the archive files are replaced. If an optional positioning character from the set **abi** is used, then the *posname* argument must be present and specifies that new files are to be placed after (**a**) or before (**b** or **i**) *posname*. Otherwise new files are placed at the end.

q Quickly append the named files to the end of the archive file. Optional positioning characters are invalid. The command does not check whether the added members are already in the archive. Useful only to avoid quadratic behavior when creating a large archive piece-by-piece.

t Print a table of contents of the archive file. If no names are given, all files in the archive are tabled. If names are given, only those files are tabled.

p Print the named files in the archive.

m Move the named files to the end of the archive. If a positioning character is present, then the *posname* argument must be present and, as in **r,** specifies where the files are to be moved.

x Extract the named files. If no names are given, all files in the archive are extracted. In neither case does **x** alter the archive file.

v Verbose. Under the verbose option, *nar* gives a file-by-file description of the making of a new archive file from the old archive and the constituent files. When used with **t,** it gives a long listing of all information about the files. When used with **p,** it precedes each file with a name.

c Create. Normally *nar* will create *afile* when it needs to. The create option suppresses the normal message that is produced when *afile* is created.

l Local. Normally *nar* places its temporary files in the directory /tmp. This option causes them to be placed in the local directory.

FILES

/tmp/v* temporaries

SEE ALSO

ar(1), arcv(1), ld(1), lorder(1), nar(5)

BUGS

If the same file is mentioned twice in an argument list, it may be put in the archive twice.

NAME

ncheck − generate names from i-numbers

SYNOPSIS

ncheck [**−i** numbers] [**−u** numbers ids] [**−a**] [filesystem]

DESCRIPTION

Ncheck with no argument generates a pathname vs. i-number list of all files on a set of default file systems. The **−i** flag reduces the report to only those files whose i-numbers follow. The **−u** option reduces the report to files owned by the specified user ids. The report produced by **−u** contains the i-number, size of the file in blocks, the name of the owner of the file (if known) and the pathname of the file. All entries are separated by a tab character to facilitate sorting by size, owner, i-node or pathname. If no numbers or ids are supplied with the **−u** or **−i** option, ncheck will read numbers or ids from the standard input. The ids or numbers may be separated by tabs, blanks or newlines. **−a** allows printing of the names '.' and '..', which are ordinarily suppressed. A file system may be specified as an argument. The raw device which references the filesystem should be specified for efficiency.

SEE ALSO

check(1M), dcheck(1M), sort(1)

BUGS

It is slow.

NAME

newgrp – log in to a new group

SYNOPSIS

newgrp [group]

DESCRIPTION

Newgrp changes the group identification of its caller, analogously to *login*(1). The same person remains logged in, and the current directory is unchanged, but calculations of access permissions to files are performed with respect to the new *group* ID.

If no argument is supplied the group in the password file for the current user is used. If the *group* specified is the same as the group in the password file, further permission checking is required. Otherwise a check is made to see if the user is present in the member list of the group file before the *newgrp* is allowed. If the user is not in the member list, and the *group* has a password, then the user is required to enter the password. In all other cases the *newgrp* attempt will fail.

When most users log in, they are members of the group named 'other.'

FILES

/etc/group, /etc/passwd

SEE ALSO

login(1), group(5)

BUGS

Since the current user's shell is replaced, any shell variables are lost.

NAME

newscheck - check to see if user has news

SYNOPSIS

newscheck [yne] [readnews options]

DESCRIPTION

newscheck reports to the user whether or not he has news.

y Reports "There is news" if the user has news to read.

n Reports "No news" if their isn't any news to read.

e Executes readnews(1) if there is news.

If there are no options, **y** is the default.

FILES

/usr/lib/news/active	Active newsgroups
~/.newsrc	Options and list of previously read articles

SEE ALSO

readnews(1), inews(1)

NAME

news − print news items

SYNOPSIS

news [**−a**] [**−g** grpname] [**−n**] [items]

DESCRIPTION

News is used to keep the user informed of current events. By convention, these events are described by files in the directory **/usr/news**.

When invoked without arguments, *news* prints the contents of all current files, owned by group "other" or owned by the user's group, in **/usr/news**, most recent first, with each preceded by an appropriate header. If the *−g* grpname option is specified (any number of times) files owned by the named groups are also printed. The group name *all* is special and means all groups, or in other words all current news items. *News* stores the "currency" time as the modification date of a file named **.news_time** in the user's home directory (the identity of this directory is determined by the environment variable **$HOME**); only files more recent than this will henceforth be considered "current."

The **−a** option causes *news* to print all items, regardless of currency. In this case, the stored time is not changed.

The **−n** option causes *news* to report the names of the current items without printing their contents, and without changing the stored time. It is useful to include such an invocation of *news* in one's **.profile** file, or in the system's **/etc/profile**.

All other arguments are assumed to be specific news items that are to be printed.

If a *delete* is typed during the printing of a news item, printing stops and the next item is started. A second *delete* within a second of the first causes the program to terminate.

FILES

/etc/profile
/usr/news/*
$HOME/.news_time

SEE ALSO

profile(5), environ(7).

NAME

nice − run a command at specified priority

SYNOPSIS

nice [−prio] command [arguments]

DESCRIPTION

Nice executes *command* at the priority specified by *prio* which is a number in the range 20 to −127. If the number is missing the default is 10. A negative number is accepted only if the user is the super-user.

SEE ALSO

nohup(1), nice(2)

BUGS

Requiring the typing of:

nice −−10 cmd

to set a negative priority is a kludge.

NAME

nm − print name list

SYNOPSIS

nm [−gnoprsuta] [file ... **]**

DESCRIPTION

Nm prints the name list (symbol table) of each object *file* in the argument list. If an argument is an archive, a listing for each object file in the archive will be produced. If no *file* is given, the symbols in **a.out** are listed.

Each symbol name is preceded by its value (blanks if undefined) and one of the letters **U** (undefined), **A** (absolute), **T** (text segment symbol), **D** (data segment symbol), **B** (bss segment symbol), **R** (register symbol), **F** (file symbol), or **C** (common symbol). If the symbol is local (non-external) the type letter is in lower case. The output is sorted alphabetically.

Options are:

−g Print only global (external) symbols.

−n Sort numerically rather than alphabetically.

−o Prepend file or archive element name to each output line rather than only once. This option can be used to make piping to *grep*(1) more meaningful.

−p Don't sort; print in symbol-table order.

−r Sort in reverse order.

−s Sort according to the size of the external symbol (computed from the difference between the value of the symbol and the value of the symbol with the next highest value). This difference is the value printed. This flag turns on **−g** and **−n** and turns off **−u** and **−p**.

−u Print only undefined symbols.

−t Print the namelist including a field that indicated which switchable text area a symbol is in. The following information is printed for each symbol: **NS** (symbol is in a NonSwitchable text area), **S0** (symbol is in Switchable text space 0), (**S1** for space1, **S2** for space2 and **S3** for space3) and **...** (not a text symbol).

−a Sort according to text space. May be used with the **−n** option to sort by address after sorting by text space. This option turns on **−t**.

SEE ALSO

ar(1), a.out(5), ar(5)

NAME

nnroff − format text

SYNOPSIS

nnroff [option] ... [file] ...

DESCRIPTION

Nnroff is the UNIX 3.0 *nroff* subsystem. It corrects several bugs in the CB-UNIX *nroff*(1) and uses newer macro packages for *mm* and *mmh*, which contain several new macros (see document referenced below). Compacted macros work correctly for *nnroff*; one can say *nnroff* −*cm* to use the compacted version of the memorandum macros *mm*. *Nnroff* will run slightly faster with compacted macros. *Nnroff* has been enhanced to work with the Virtual Terminal Protocol (see *vtp*(4)); *nnroff* will accept the terminal option −**Tvirtual**. See *nroff*(1) and the document below for *options* available.

SEE ALSO

nroff(1), vtp(4)

N. E. Bock, *Changes to Text Processing Software for UNIX Release 3.0* 3646-800407.01MF PY April 7, 1980

NAME

nohup – run a command immune to hangups

SYNOPSIS

nohup command [arguments]

DESCRIPTION

Nohup executes *command* with hangups, interrupts, and quits ignored. If output is not redirected by the user, it will be sent to **nohup.out**.

SEE ALSO

nice(1), signal(2)

NAME

nroff, troff − format or typeset text

SYNOPSIS

nroff [option] ... [file] ...
troff [option] ... [file] ...

DESCRIPTION

Nroff formats text in the named *file*s (standard input by default) for printing on typewriter-like devices; similarly, *troff* formats text for a Wang-Graphic Systems, Inc. C/A/T phototypesetter. Their capabilities are described in the first manual cited below.

An argument consisting of a single minus (−) is taken to be a file name corresponding to the standard input. The options, which may appear in any order so long as they appear before the files, are:

−o*list* Print only pages whose page numbers appear in the comma-separated *list* of numbers and ranges. A range *N*−*M* means pages *N* through *M*; an initial −*N* means from the beginning to page *N*; and a final *N*− means from *N* to the end.

−n*N* Number first generated page *N*.

−s*N* Stop every *N* pages. *Nroff* will halt prior to every *N* pages (default *N*=1) to allow paper loading or changing, and will resume upon receipt of a new-line. *Troff* will stop the phototypesetter every *N* pages, produce a trailer to allow changing cassettes, and resume when the typesetter's start button is pressed.

−m*name* Prepend the macro file **/usr/lib/tmac/tmac.***name* to the input *files*.

−r*aN* Set register *a* (which has a one-character name) to *N*.

−**i** Read standard input after the input files are exhausted.

−**q** Invoke the simultaneous input-output mode of the **.rd** request.

Nroff only

−T*name* Prepare output for specified terminal. Known *name*s are **37** for the (default) TELETYPE® Model 37 terminal, **tn300** for the GE TermiNet 300 (or any terminal without half-line capability), **300s** for the DASI 300s, **300** for the DASI 300, **450** for the DASI 450, and **lp** for a (generic) line printer.

−**e** Produce equally-spaced words in adjusted lines, using the full resolution of the particular terminal.

−**h** Use output tabs during horizontal spacing to speed output and reduce output character count. Tab settings are assumed to be every 8 nominal character widths.

Troff only

−**t** Direct output to the standard output instead of the phototypesetter.

−**f** Refrain from feeding out paper and stopping phototypesetter at the end of the run.

−**w** Wait until phototypesetter is available, if currently busy.

−**b** Report whether the phototypesetter is busy or available. No text processing is done.

−**a** Send a printable ASCII approximation of the results to the standard output.

−pN Print all characters in point size *N* while retaining all prescribed spacings and motions, to reduce phototypesetter elapsed time.

−**g** Prepare output for the Murray Hill Computation Center phototypesetter and direct it to the standard output (see *gcat*(1C)).

FILES

/usr/lib/suftab	suffix hyphenation tables
/tmp/ta$#	temporary file
/usr/lib/tmac/tmac.*	standard macro file pointers
/usr/lib/macros/*	standard macro files
/usr/lib/term/*	terminal driving tables for *nroff*
/usr/lib/font/*	font width tables for *troff*

SEE ALSO

NROFF/TROFF User's Manual by J. F. Ossanna
A TROFF Tutorial by B. W. Kernighan
eqn(1), tbl(1),
col(1), mm(1) (*nroff* only)
gcat(1C), tc(1) (*troff* only).
nnroff(1)

NAME

occ − old C compiler

SYNOPSIS

occ [**−c**] [**−p**] [**−f**] [**−O**] [**−S**] [**−P**] [**−E**] [**−Dsymbol** ...] [**−Usymbol** ...] [**−Iprefix**] [**−C**] file ...

DESCRIPTION

Occ is functionally identical to the C compiler which was supplied with CB UNIX Release 1. Its options and actions are identical to *cc*(1). The default libraries which it assumes are those which provide a user interface almost identical to that provided in Release 1. An executable module produced using *occ* will be version stamped so that an appropriate system interface is provided by the operating system.

The intent of supplying *occ* is that existing modules may continue to be compiled and executed in the environment it provides, allowing gradual conversion of all code to use *cc*. All new code should be written using *cc*.

FILES

/lib/oc[01]	compiler
/lib/oc2	optional optimizer
/lib/crt0.o	runtime startoff
/lib/mcrt0.o	runtime startoff of profiling
/lib/liboc.a	C library
/lib/liboa.a	Assembler library
/lib/liboS.a	Standard I/O library; see *stdio:o(3S)*

SEE ALSO

cc(1), stamp(1), ostdio(3S), intro(2), monitor(3), prof(1), ld(1),
Programming in C— a tutorial,
C Reference Manual.

NAME

od − octal dump

SYNOPSIS

od [**-abcdho**] [file] [[+] offset [.][b]]

DESCRIPTION

Od dumps *file* in one or more formats as selected by the first argument. If the first argument is missing -o is default. The meanings of the format argument characters are:

a interprets words as PDP-11 instructions and dis-assembles the operation code. Unknown operation codes print as ???.

b interprets bytes in octal.

c interprets bytes in ASCII. Unknown ASCII characters are printed as ?.

d interprets words in decimanl.

h interprets words in hex.

o interprets words in octal.

The *file* argument specifies which file is to be dumped. If no *file* argument is specified, the standard input is used. Thus *od* can be used as a filter.

The *offset* argument specifies the offset in the file where dumping is to commence. This argument is normally interpreted as octal bytes. If '.' is appended, *offset* is interpreted in decimal. If 'b' is appended, *offset* is interpreted in blocks (A block is 512 bytes). If *file* is omitted, *offset* must be preceded by '+'.

Dumping continues until end-of-file.

SEE ALSO

adb(1)

NAME

over — overstrike optimizer

SYNOPSIS

over

DESCRIPTION

Over reads the standard input and writes the standard output. It interprets and then removes backspaces, constructing line buffers which contain only forward motions and are separated by carriage returns. This greatly reduces print time for printing devices incapable of backspaces (such as LP11 printers). *Over* is generally used to filter the output produced by *nroff*.

SEE ALSO

nroff(1), col(1)

BUGS

Maximum line length is 500 characters. Won't handle more than 10 overstrikes. *Over* should be built into the output device as needed; the same comment applies to *col*.

NAME

pack − compress files

SYNOPSIS

pack [−] name ...

DESCRIPTION

Pack attempts to store the specified files in a compressed form. Wherever possible (and useful), each input file *name* is replaced by a packed file *name*.**z** with the same access modes, access and modified dates, and owner as those of *name*. If *pack* is successful, *name* will be removed. Packed files can be restored to their original form using *unpack*(1) or *pcat*(1).

Pack uses Huffman (minimum redundancy) codes on a byte-by-byte basis. If the − argument is used, an internal flag is set that causes the number of times each byte is used, its relative frequency, and the code for the byte to be printed on the standard output. Additional occurrences of − in place of *name* will cause the internal flag to be set and reset.

The amount of compression obtained depends on the size of the input file and the character frequency distribution. Because a decoding tree forms the first part of each .z file, it is usually not worthwhile to pack files smaller than three blocks, unless the character frequency distribution is very skewed, which may occur with printer plots or pictures.

Typically, text files are reduced to 60-75% of their original size. Load modules, which use a larger character set and have a more uniform distribution of characters, show little compression, the packed versions being about 90% of the original size.

Pack returns a value that is the number of files that it failed to compress. No packing will occur if:

the file appears to be already packed;
the file name has more than 12 characters;
the file has links;
the file is a directory;
the file cannot be opened;
no disk storage blocks will be saved by packing;
a file called *name*.**z** already exists;
the .**z** file cannot be created;
an I/O error occurred during processing.

The last segment of the file name must contain no more than 12 characters to allow space for the appended .**z** extension. Directories cannot (and should not) be compressed.

SEE ALSO

pcat(1), unpack(1).

NAME

padm − program administration system

SYNOPSIS

padm cmd [options] [files]

DESCRIPTION

Padm is a collection of shell programs which will assist a project in using the Source Code Control System (SCCS). Virtually all of the commands begin with the letter 'g' and many of them are just 'g' prepended to the appropriate SCCS command.

The Padm allows a group of people to define a directory which subtends all SCCS source directories of interest. The shell variable SCCSOURCE is then set to this directory and *exported*. The user then defines SUBSYSTEMS which are of interest. Specifically, the shell variable SUBSYSTEMS is set to the directories relative to $SCCSOURCE which are of interest.

As an example assume a project's SCCS files are all under the directory /usr/proect. If one wants to work on a specific subsystem in "project" one must do the following:

```
SCCSOURCE=/usr/project
SUBSYSTEMS="dispatcher watchdog"
export SCCSOURCE SUBSYSTEMS
```

This would imply the directories /usr/project/dispatcher and /usr/project/watchdog exist and contain SCCS files or directories. Assuming the directories contain SCCS files one may *gget* or *gdelta* any file in either of the two directories. One need only refer to a file by its non-SCCS filename. Thus if in /usr/project/dispatcher there is a file called s.startup.c he can get for the purpose of editing s.startup.c by doing the following:

```
gget -e startup.c
```

In addition if one wanted a particular verson one could say:

```
gget -r3.4 startup.c
```

which would retrieve SCCS version 3.4 of s.startup.c in /usr/project/dispatcher. (This time no edit is implied.) In addition if one wanted to find out what versions existed one could type:

```
gprt startup.c
```

Likewise, when it comes time to put the source code back into SCCS one types:

```
gdelta startup.c
```

The delta(1S) command will respond with normal run of the mill delta stuff ("comments?" and "MR'S?" if the flag is set) and perform the delta.

The current version of the Padm allows the user to get a file no matter what directory one is in. This might present a problem if more than one directory is specified in $SUBSYSTEMS. For instance, if both /usr/project/dispatcher and /usr/project/watchdog have a 'Makefile' and the user's environment is:

```
SCCSOURCE=/usr/project
SUBSYSTEMS="dispatcher watchdog"
```

and the user types:

gget -e Makefile

the user will get the 'Makefile' from the first directory listed in $SUBSYSTEMS. To get the 'Makefile' from watchdog the user can type:

gget watchdog/Makefile

However, when one *delta*'s the file 'Makefile' if one does not specify watchdog/Makefile one will get an error (unless one is also editing the dispatcher/Makefile). Hence, under certain circumstances one could *delta* a 'Makefile' into the wrong spot. This problem could be avoided by allowing only one directory entry in $SUBSYSTEMS. However, at this time, we have had not had this problem so no attempt has been made to solve it.

When a subsystem is "finished" (?) it is assumed that the programmer wants to *mark* it in some way such that two months later when one has fixed a bug one can know what it was that went to the field. For this purpose, the *gmark*(1S) command exists. One can type:

gmark subsys

and the latest release of every SCCS file in subsys is remembered in an SCCS file originated and maintained completely by *gmark*(1S). Thus, if one were working on the shell in /usr/project/sh and wanted to mark the shell one would type:

gmark sh

The mark file would contain the name and version of each SCCS file in /usr/project/sh. To see this one could type:

gmark -L sh

which gives a partially formatted list of files with their versions. Note, the markfile is maintained in SCCS format and the only interface to it is through the *gmark*(1S) command. When making a new markfile the *gmark* command *get*'s for the purpose of editing the markfile; notes the most recent versions (with a 'get -g .') of each SCCS file in the named directory (excluding the markfile itself); and *delta*'s this new information into the markfile. The '-L' command just does a 'get -p' on the markfile.

In addition, when one *gmark*'s a directory, all subtending directories are marked. Also, with each mark the user must supply a comment which goes into each new delta'ed markfile.

For further, information on the various commands one is urged to read the individual manual pages for the following commands: gget(1S), gdelta(1S), gadd(1S), gmark(1S), gprt(1S) and gadmin(1S). Also note the convention of upper case letters for options meant specifically for the padm commands. This is an attempt to avoid collisions with SCCS options which at present are all lower case letters.

NAME

passwd – change login password

SYNOPSIS

passwd name

DESCRIPTION

This command changes (or installs) a password associated with the login *name*.

The program prompts for the old password (if any) and then for the new one (twice). The caller must supply these. New passwords should be at least four characters long if they use a sufficiently rich alphabet and at least six characters long if monocase. Only the first eight characters of the password are significant.

Only the owner of the name or the super-user may change a password; the owner must prove he knows the old password. Only the super-user can create a null password.

The password file is not changed if the new password is the same as the old password, or if the password has not "aged" sufficiently. (See *passwd*(5)).

FILES

/etc/passwd

SEE ALSO

login(1), passwd(5), crypt(3C)

NAME

paste − merge same lines of several files or subsequent lines of one file

SYNOPSIS

paste file1 file2 ...
paste **−d**list file1 file2 ...
paste **−s** [**−d**list] file1 file2 ...

DESCRIPTION

In the first two forms, *paste* concatenates corresponding lines of the given input files *file1, file2* etc. It treats each file as a column or columns of a table and pastes them together horizontally (parallel merging). If you will, it is the counterpart of *cat*(1) which concatenates vertically, i.e. one file after the other. In the last form above, *paste* subsumes the function of an older command with the same name by combining subsequent lines of the input file (serial merging). In all cases, lines are glued together with the *tab* character, or with characters from an optionally specified *list*. Output is to the standard output, so it can be used as the start of a pipe, or as a filter, if − is used in place of a filename.

The meanings of the options are:

−d Without this option, the new-line characters of each but the last file (or last line in case of the **−s** option) are replaced by a *tab* character. This option allows replacing the *tab* character by one or more alternate characters (see below).

list One or more characters immediately following **−d** replace the default *tab* as the line concatenation character. The list is used circularly, i. e. when exhausted, it is reused. In parallel merging (i. e. no **−s** option), the lines from the last file are always terminated with a new-line character, not from the *list*. The list may contain the special escape sequences: \n (new-line), \t (tab), \\ (backslash), \0 (empty string, not a null character). Quoting may be necessary, if characters have special meaning to the shell (e.g. to get one backslash, write "" *−d*"\\\\").

−s Merge subsequent lines rather than one from each input file. Use *tab* for concatenation, unless a *list* is specified with **−d** option. Regardless of the *list*, the very last character of the file is forced to be a new-line.

− May be used in place of any filename, to read a line from the standard input. (There is no prompting).

EXAMPLES

ls | paste −d" " −	list directory in one column
ls | paste − − − −	list directory in four columns
paste −s −d"\t\n" file	combine pairs of lines into lines

SEE ALSO

grep(1), cut(1), pr(1): *pr −t −m* ... works similarly, but creates extra blanks, tabs and new-lines for a nice page layout.

DIAGNOSTICS

line too long: Output lines are restricted to 256 characters.
too many files: Except for **−s** option, no more than 12 input files may be specified.