NAME

lint − a C program verifier

SYNOPSIS

lint [**−abchmnpuvx**] file ...

DESCRIPTION

Lint attempts to detect features of the C program *files* which are likely to be bugs, or non-portable, or wasteful. It also checks the type usage of the program more strictly than the compilers. Among the things which are currently found are unreachable statements, loops not entered at the top, automatic variables declared and not used, and logical expressions whose value is constant. Moreover, the usage of functions is checked to find functions which return values in some places and not in others, functions called with varying numbers of arguments, and functions whose values are not used.

By default, it is assumed that all the *files* are to be linked together; they are checked for mutual compatibility. Function definitions for the standard C and UNIX system libraries are available to *lint* by default. A subset of the standard C library is used when *lint* is invoked with the **−p** option.

Any number of the options in the following list may be used. The **−D**, **−U**, and **−I** options of *cc*(1) are also recognized as separate arguments.

−p Attempt to check portability to the IBM and GCOS dialects of C.

−h Apply a number of heuristic tests to attempt to intuit bugs, improve style, and reduce waste.

−b Report **break** statements that cannot be reached. (This is not the default because, unfortunately, most *lex* and many *yacc* outputs produce dozens of such comments.)

−v Suppress complaints about unused arguments in functions.

−x Report variables referred to by extern declarations, but never used.

−a Report assignments of long values to int variables.

−c Complain about casts which have questionable portability.

−u Do not complain about functions and variables used and not defined, or defined and not used (this is suitable for running *lint* on a subset of files out of a larger program).

−n Do not check compatibility against the standard library.

−m Do not complain about user-defined functions that have the same name as functions in a referenced library. (The difference between this and the previous option is that the **−m** option will still allow cmpatibility of usage to be checked for library functions not redefined by the user.)

Exit(2) and other functions which do not return are not understood; this causes various lies.

Certain conventional comments in the C source will change the behavior of *lint*:

/*NOTREACHED*/
at appropriate points stops comments about unreachable code.

/*VARARGS*n**/
suppresses the usual checking for variable numbers of arguments in the following function declaration. The data types of the first *n* arguments are checked; a missing *n* is taken to be 0.

/*NOSTRICT*/
shuts off strict type checking in the next expression.

/*ARGSUSED*/

turns on the −v option for the next function.

/*LINTLIBRARY*/
: at the beginning of a file shuts off complaints about unused functions in this file.

FILES

/usr/lib/lint[12]	programs
/usr/lib/llib-lc	declarations for standard functions
/usr/lib/llib-port	declarations for portable functions
/usr/tmp/lint.*	temporaries

SEE ALSO

cc(1).

NAME

load − load UNIX object modules

SYNOPSIS

load [−D] [−s] [−K[p]] [version] [file [...]]

DESCRIPTION

Load will build a specified version of UNIX by invoking the link editor *ld*(1) on the modules

/usr/src/ucb/os/low.*version*.o,
/usr/src/ucb/os/mch.*version*.o,
/usr/src/ucb/os/conf.*version*.o,
/usr/src/ucb/os/lib1.*version*.a, and
/usr/src/ucb/io/lib2.*version*.a.

The resulting module is a loadable UNIX and is placed in the file unix.*version* in the current directory. A list of the sizes of the new module, the corresponding version in **/util**, and the current **/unix** is produced.

If any object modules are specified as file arguments, these are loaded in the appropriate order in preference to the designated versions. This allows a test version of the system to be readily built without having to place the tested modules in the system libraries.

If no version is specified, the default is **70**. Thus, the command *load* by itself will produce a UNIX.70 from the current source modules and libraries.

Optional flags consist of:

−D default; causes the default modules lib*x*.*70*.a or module.*70*.o to be used if no corresponding module can be found for the specified version.

−s silent; do not echo commands being executed or produce the size comparisons.

−K build a **UNIX** that allows a text size greater than 65536 bytes. The optional *p* parameter is the number of kernel memory management registers that do not change when switching text areas. *P* should be in the range from 1 through 7 with a default value of 7.

NAME

login − sign on

SYNOPSIS

login [user environment−variables]

DESCRIPTION

The *login* command is used at the beginning of each terminal session and allows you to identify yourself to the system. It may be invoked explicitly as a command, and is invoked by the system when a connection is first established, or after the previous user has logged out by sending an "end-of-file" (control−D) to his or her initial shell. (See *How to Get Started* at the beginning of this volume for instructions on how to dial up initially.)

Login asks for your user name (if not supplied as an argument), and, if appropriate, your password. Echoing is turned off (where possible) during the typing of your password, so it will not appear on the written record of the session.

At some installations, an option may be invoked that will require you to enter a second "dialup" password. This will occur only for dial-up connections, and will be prompted by the message "dialup password:". Both passwords are required for a successful login.

If you do not complete the login successfully within a certain period of time (e.g., one minute), you are likely to be silently disconnected.

After a successful login, accounting files are updated, the message-of-the-day, if any, is printed. *Login* initializes the user and group IDs and the working directory, then executes a command interpreter (usually *sh*(1)) according to specifications found in the /etc/passwd file. Argument 0 of the command interpreter is − followed by the last component of the interpreter's pathname.

The basic *environment* (see *environ*(7)) is initialized to:

```
HOME=your-login-directory
PATH=:/bin:/usr/bin
SHELL=last−field−of−password−entry
MAIL=/usr/mail/your−login−name
TZ=timezone specification
```

The environment may be expanded or modified by supplying additional arguments to *login*, either at execution time or when *login* requests your login name. The arguments may take either the form *xxx* or *xxx=yyy*. Arguments without an equal sign in them are placed in the environment as

L*n*=xxx

where *n* is a number starting at 0 and is incremented each time a new variable name is required. Variables containing an '=' are placed into the environment without modification. If they already appear in the environment, then they replace the older value. There are two exceptions. The variables **PATH** and **SHELL** cannot be changed. This prevents people logging into restricted shell environments from spawning secondary shells which aren't restricted. Both *login* and *getty* understand simple single character quoting conventions. Typing a backslash in front of a character quotes it and allows the inclusion of such things as spaces and tabs.

FILES

/etc/utmp	accounting
/usr/adm/wtmp	accounting
/usr/mail/your-name	mailbox for user *name*
/etc/motd	message-of-the-day
/etc/passwd	password file

SEE ALSO

getty(1M), mail(1), newgrp(1), passwd(1), sh(1), su(1), passwd(5), profile(5), environ(7).

DIAGNOSTICS

"Login incorrect" if the user name or the password is incorrect.
"No shell", "cannot open password file", "no directory": consult a UNIX programming counselor.

NAME
lorder − find ordering relation for an object library

SYNOPSIS
lorder file ...

DESCRIPTION
The input is one or more object or library archive (see *ar*(1)) *files.* The standard output is a list of pairs of object file names, meaning that the first file of the pair refers to external identifiers defined in the second. The output may be processed by *tsort*(1) to find an ordering of a library suitable for one-pass access by *ld*(1).

This brash one-liner intends to build a new library from existing '.o' files.

```
ar cr library `lorder *.o | tsort`
```

FILES

*symref, *symdef	temp files

SEE ALSO
ar(1), ld(1), tsort(1)

BUGS
Object files whose name do not end with '.o', even when contained in library archives, are overlooked. Their global symbols and references are attributed to some other file.

NAME

lpr − line printer spooling program

SYNOPSIS

lpr [queue] [[+/−] [[+/−]file [section]]

DESCRIPTION

Lpr arranges to have the line printer demon print the specified files on one of the printers servicing *queue*. If *queue* is not specified, *lpr* will search for a default queue on which to place the request.

Normally each file is printed in the state in which it is found by the demon. If a + option has been set, or immediately precedes the file, then *lpr* makes a copy for the demon to print. In a like manner, if the option − is set, then lpr will remove the source file. If a file is followed by a *section*, in the form 'begin,end' (no spaces), *lpr* arranges for only a section of the file to be printed, beginning with the line numbered *begin*, and ending with the line numbered *end*.

If there are no file arguments, then *lpr* reads its standard input. After all files have been queued or EOF has been detected on standard input, *lpr* assigns a job id which may be used with the various queue control commands.

FILES

/usr/lpd/*	spooling area
/etc/passwd	user ids
/usr/lpd/.printers	defines printers and printer options
/usr/lpd/.qmap	defines queues and printer-to-queue maps
/etc/lpd	printer demon

SEE ALSO

abort(1), hold(1), init(1), release(1), restrain(1), start(1)

NAME

ls − list contents of directory

SYNOPSIS

ls [**−ltasdrucif**] name ...

DESCRIPTION

For each directory argument, *ls* lists the contents of the directory; for each file argument, *ls* repeats its name and any other information requested. The output is sorted alphabetically by default. When no argument is given, the current directory is listed. When several arguments are given, the arguments are first sorted appropriately, but file arguments appear before directories and their contents. There are several options:

−l	List in long format, giving mode, number of links, owner, group, size in bytes, and time of last modification for each file. (See below.) If the file is a special file the size field will instead contain the major and minor device numbers.
−t	Sort by time modified (latest first) instead of by name, as is normal.
−a	List all entries; usually those beginning with **.** are suppressed.
−s	Give size in blocks, including indirect blocks, for each entry.
−d	If argument is a directory, list only its name, not its contents (mostly used with **−l** to get status on directory).
−r	Reverse the order of sort to get reverse alphabetic or oldest first as appropriate.
−u	Use time of last access instead of last modification for sorting (**−t**) or printing (**−l**).
−c	Use time of last modification to inode (mode, etc.) instead of last modification to file for sorting (**−t**) or printing (**−l**).
−i	Print i-number in first column of the report for each file listed.
−f	Force each argument to be interpreted as a directory and list the name found in each slot. This option turns off **−l**, **−t**, **−s**, and **−r**, and turns on **−a**; the order is the order in which entries appear in the directory.

The mode printed under the **−l** option contains 11 characters which are interpreted as follows:

The first character is:

d	if the entry is a directory;
b	if the entry is a block-type special file;
c	if the entry is a character-type special file;
m	if the entry is a multiplexed character file;
n	if the entry is a multiplexed block file;
−	if the entry is a plain file.

The next 9 characters are interpreted as three sets of three bits each. The first set refers to owner permissions; the next to permissions to others in the same user-group; and the last to all others. Within each set the three characters indicate permission respectively to read, to write, or to execute the file as a program. For a directory, 'execute' permission is interpreted to mean permission to search the directory for a specified file. The permissions are indicated as follows:

r	if the file is readable;
w	if the file is writable;
x	if the file is executable;
−	if the indicated permission is not granted.

The group-execute permission character is given as s if the file has set-group-ID mode and group-execute permission; likewise the user-execute permission character is given as s if the file has set-user-ID mode and user-execute permission. If the set-user-ID bit or set-group-ID bit is set but the file does not have the appropriate execute permission, a ? will appear where the s normally would be seen.

Similarly, the other-execute permission character may be replaced by a **t** if the save-text and other-execute bits are both set. If only the save-text bit is set a ? will appear instead.

FILES

/etc/passwd and /etc/group to get user and group ID's for **ls −l**.

NAME

ls - list contents of directory

SYNOPSIS

ls [**-abcdfgilmqrstux1CFR**] name ...
l [ls options] name ...

DESCRIPTION

For each directory argument, **ls** lists the contents of the directory; for each file argument, **ls** repeats its name and any other information requested. The output is sorted alphabetically by default. When no argument is given, the current directory is listed. When several arguments are given, the arguments are first sorted appropriately, but file arguments appear before directories and their contents.

There are three major listing formats. The format chosen depends on whether the output is going to a teletype, and may also be controlled by option flags. The default format for a teletype is to list the contents of directories in multi-column format, with the entries sorted down the columns. (Files which are not the contents of a directory being interpreted are always sorted across the page rather than down the page in columns. This is because the individual file names may be arbitrarily long.) If the standard output is not a teletype, the default format is to list one entry per line. Finally, there is a stream output format in which files are listed across the page, separated by `,' characters. The **-m** flag enables this format; when invoked as l this format is also used.

There are an unbelievable number of options:

-l List in long format, giving mode, number of links, owner, group, size in bytes, and time of last modification for each file. (See below.) If the file is a special file the size field will instead contain the major and minor device numbers.

-t Sort by time modified (latest first) instead of by name, as is normal.

-a List all entries; usually `.' and `..' are suppressed.

-s Give size in blocks, including indirect blocks, for each entry.

-d If argument is a directory, list only its name, not its contents (mostly used with **-l** to get status on directory).

-r Reverse the order of sort to get reverse alphabetic or oldest first as appropriate.

-u Use time of last access instead of last modification for sorting (**-t**) or printing (**-l**).

-c Use time of file creation for sorting or printing.

-i Print i-number in first column of the report for each file listed.

-f Force each argument to be interpreted as a directory and list the name found in each slot. This option turns off **-l**, **-t**, **-s**, and **-r**, and turns on **-a**; the order is the order in which entries appear in the directory.

-g Give group ID instead of owner ID in long listing.

-m force stream output format

-1 force one entry per line output format, e.g. to a teletype

-C force multi-column output, e.g. to a file or a pipe

-q force printing of non-graphic characters in file names as the character `?'; this normally happens only if the output device is a teletype

-b force printing of non-graphic characters to be in the \ddd notation, in octal.

-x force columnar printing to be sorted across rather than down the page; this is the default if the last character of the name the program is invoked with is an `x'.

-F cause directories to be marked with a trailing `/' and executable files to be marked with a trailing `*'; this is the default if the last character of the name the program is invoked with is a `f'.

-R recursively list subdirectories encountered.

The mode printed under the **-l** option contains 11 characters which are interpreted as follows: the first character is

d if the entry is a directory;
b if the entry is a block-type special file;
c if the entry is a character-type special file;
m if the entry is a multiplexor-type character special file;
- if the entry is a plain file.

The next 9 characters are interpreted as three sets of three bits each. The first set refers to owner permissions; the next to permissions to others in the same user-group; and the last to all others. Within each set the three characters indicate permission

respectively to read, to write, or to execute the file as a program. For a directory, `execute' permission is interpreted to mean permission to search the directory for a specified file. The permissions are indicated as follows:

r	if the file is readable;
w	if the file is writable;
x	if the file is executable;
-	if the indicated permission is not granted.

The group-execute permission character is given as **s** if the file has set-group-ID mode; likewise the user-execute permission character is given as **s** if the file has set-user-ID mode.

The last character of the mode (normally `x' or `-') is **t** if the 1000 bit of the mode is on. See _chmod_(1) for the meaning of this mode.

When the sizes of the files in a directory are listed, a total count of blocks, including indirect blocks is printed.

FILES

/etc/passwd to get user ID's for `ls -l'.
/etc/group to get group ID's for `ls -g'.

BUGS

Newline and tab are considered printing characters in file names.

The output device is assumed to be 80 columns wide.

The option setting based on whether the output is a teletype is undesirable as ``ls -s'' is much different than ``ls -s | lpr''. On the other hand, not doing this setting would make old shell scripts which used _ls_ almost certain losers.

Column widths choices are poor for terminals which can tab.

NAME

m4 – macro processor

SYNOPSIS

m4 [files]

DESCRIPTION

M4 is a macro processor intended as a front end for Ratfor, C, and other languages. Each of the argument files is processed in order; if there are no arguments, or if an argument is –, the standard input is read. The processed text is written on the standard output.

Macro calls have the form

```
name(arg1,arg2, . . . , argn)
```

The (must immediately follow the name of the macro. If a defined macro name is not followed by a (, it is deemed to have no arguments. Leading unquoted blanks, tabs, and newlines are ignored while collecting arguments. Potential macro names consist of alphabetic letters, digits, and underscore _, where the first character is not a digit.

Left and right single quotes are used to quote strings. The value of a quoted string is the string stripped of the quotes.

When a macro name is recognized, its arguments are collected by searching for a matching right parenthesis. Macro evaluation proceeds normally during the collection of the arguments, and any commas or right parentheses which happen to turn up within the value of a nested call are as effective as those in the original input text. After argument collection, the value of the macro is pushed back onto the input stream and rescanned.

M4 makes available the following built-in macros. They may be redefined, but once this is done the original meaning is lost. Their values are null unless otherwise stated.

define	The second argument is installed as the value of the macro whose name is the first argument. Each occurrence of $*n* in the replacement text, where *n* is a digit, is replaced by the *n*-th argument. Argument 0 is the name of the macro; missing arguments are replaced by the null string.
undefine	removes the definition of the macro named in its argument.
ifdef	If the first argument is defined, the value is the second argument, otherwise the third. If there is no third argument, the value is null. The word *unix* is predefined on UNIX versions of *m4*.
changequote	Change quote characters to the first and second arguments. *Changequote* without arguments restores the original values (i.e., \`').
divert	*M4* maintains 10 output streams, numbered 0–9. The final output is the concatenation of the streams in numerical order; initially stream 0 is the current stream. The *divert* macro changes the current output stream to its (digit-string) argument. Output diverted to a stream other than 0 through 9 is discarded.
undivert	causes immediate output of text from diversions named as arguments, or all diversions if no argument. Text may be undiverted into another diversion. Undiverting discards the diverted text.
divnum	returns the value of the current output stream.
dnl	reads and discards characters up to and including the next newline.
ifelse	has three or more arguments. If the first argument is the same string as the second, then the value is the third argument. If not, and if there are more than four arguments, the process is repeated with arguments 4, 5, 6 and 7. Otherwise, the value is either the fourth string, or, if it is not present, null.

incr	returns the value of its argument incremented by 1. The value of the argument is calculated by interpreting an initial digit-string as a decimal number.
eval	evaluates its argument as an arithmetic expression, using 32-bit arithmetic. Operators include +, −, *, /, %, ^ (exponentiation); relationals; parentheses.
len	returns the number of characters in its argument.
index	returns the position in its first argument where the second argument begins (zero origin), or −1 if the second argument does not occur.
substr	returns a substring of its first argument. The second argument is a zero origin number selecting the first character; the third argument indicates the length of the substring. A missing third argument is taken to be large enough to extend to the end of the first string.
translit	transliterates the characters in its first argument from the set given by the second argument to the set given by the third. No abbreviations are permitted.
include	returns the contents of the file named in the argument.
sinclude	is identical to *include,* except that it says nothing if the file is inaccessible.
syscmd	executes the UNIX command given in the first argument. No value is returned.
maketemp	fills in a string of XXXXX in its argument with the current process id.
errprint	prints its argument on the diagnostic output file.
dumpdef	prints current names and definitions, for the named items, or for all if no arguments are given.

SEE ALSO

B. W. Kernighan and D. M. Ritchie, *The M4 Macro Processor*, Bell Laboratories CSTR #59, 1977.

NAME

mail, rmail — send mail to users or read mail

SYNOPSIS

mail [**−rpq**] [**−f** file]

mail [**−g** group] persons

rmail persons

DESCRIPTION

Mail without arguments prints a user's mail, message-by-message, in last-in, first-out order. For each message, the user is prompted with a **?**, and a line is read from the standard input to determine the disposition of the message:

<new-line>	Go on to next message.
+	Same as <new-line>.
d	Delete message and go on to next message.
p	Print message again.
−	Go back to previous message.
s [*files*]	Save message in the named *files* (**$HOME/mbox** is default).
a	Answer a message and delete the current letter.
as [*files*]	Answer a message, delete the current letter, and save the letter and answer in *files* (**$HOME/mbox** default).
w [*files*]	Save message, without a header, in the named *files* (**$HOME/mbox** is default).
m [−**g** *groups*] [*persons*]	Mail the message to the named *persons* (yourself is default). A "Forwarded by ..." message is inserted after the header.
q	Put undeleted mail back in the *mailfile* and stop.
EOT (control-d)	Same as **q**.
x	Put all mail back in the *mailfile* unchanged and stop.
!*command*	Escape to the shell to do *command*.
*	Print a command summary.

The optional arguments alter the printing of the mail.

−r	causes messages to be printed in first-in, first-out order.
−p	causes all mail to be printed without prompting for disposition.
−q	causes *mail* to terminate after interrupts. Normally an interrupt only causes the termination of the message being printed.
−f *file*	causes *mail* to use *file* (e.g., **mbox**) instead of the default *mailfile*.
−g *group*	causes mail to be sent to members of *group* that are designated in the file /etc/**group**.

When *persons* or *groups* are named, *mail* takes the standard input up to an end-of-file (or up to a line consisting of just a '.') and adds it to each *person*'s of member's of the named *group mailfile*. The message is preceded by the sender's name and a postmark. Lines that look like postmarks in the message, (i.e., "From ...") are prepended with '>'. A *person* is usually a user name recognized by *login*(1). If a *person* being sent mail is not recognized, or if *mail* is interrupted during input, the *dead.letter* will be saved to allow editing and resending.

To denote a recipient on a remote system, prefix *person* by the system name and exclamation mark (see *uucp*(1C)). Everything after the first exclamation mark in *persons* is interpreted by the remote system. In particular, if *persons* contains additional exclamation marks, it can denote a sequence of machines through which the message is to be sent on the way to its ultimate destination. For example, specifying **a!b!cde** as a recipient's name causes the message to be sent to user **b!cde** on system **a**. System **a** will interpret that destination as a request to send the message to user **cde** on system **b**. This might be useful, for instance, if the sending system can access system **a** but not system **b**, and system **a** has access to system **b**.

The *mailfile* may be manipulated in two ways to alter the function of *mail*. The *other* permissions of the file may be read-write, read-only, or neither read nor write to allow different levels of privacy. If changed to other than the default, the file will be preserved even when empty to perpetuate the desired permissions. The file may also contain the first line:

Forward to *person*

which will cause all mail sent to the owner of the *mailfile* to be forwarded to *person*. This is especially useful to forward all of a person's mail to one machine in a multiple machine environment.

Rmail only permits the sending of mail. *Uucp*(1C) uses *rmail* as a security precaution.

When a user logs in he is informed of the presence of mail, if any.

FILES

/etc/passwd	to identify sender and locate persons
/etc/group	to identify members of groups
/usr/mail/*	incoming mail for user *
$HOME/mbox	saved mail
$MAIL	*mailfile*
/tmp/ma*	temp file
/usr/mail/*.lock	lock for mail directory
dead.letter	unmailable text

SEE ALSO

login(1), uucp(1C), write(1).

BUGS

Race conditions sometimes result in a failure to remove a lock file.
After an interrupt, the next message may not be printed, printing may be forced by responding **p**.
Mail does not handle mail items greater than 65535 characters long. The *mailfile* has a tendency to go berserk.

NAME

Mail - send and receive mail

SYNOPSIS

Mail [**-f** [name]] [people ...]

INTRODUCTION

Mail is a intelligent mail processing system, which has a command syntax reminiscent of *ed* with lines replaced by messages.

Sending mail. To send a message to one or more other people, *Mail* can be invoked with arguments which are the names of people to send to. You are then expected to type in your message, followed by an EOT (control-D) at the beginning of a line. The section below, labeled *Replying to or originating mail,* describes some features of *Mail* available to help you compose your letter.

Reading mail. In normal usage, *Mail* is given no arguments and checks your mail out of the post office, then printing out a one line header of each message there. The current message is initially the first message (numbered 1) and can be printed using the **print** command (which can be abbreviated **p**). You can move among the messages much as you move between lines in *ed,* with the commands `+' and `-' moving backwards and forwards, and simple numbers typing the addressed message.

Disposing of mail. After examining a message you can **delete** (**d**) the message or **reply** (**r**) to it. Deletion causes the *Mail* program to forget about the message. This is not irreversible, the message can be **undeleted** (**u**) by giving its number, or the *Mail* session can be aborted by giving the **exit** (**x**) command. Deleted messages will, however, usually disappear never to be seen again.

Specifying messages. Commands such as **print** and **delete** often can be given a list of message numbers as argument to apply to a number of messages at once. Thus ``delete 1 2'' deletes messages 1 and 2, while ``delete 1-5'' deletes messages 1 through 5. The special name ``*'' addresses all messages, and ``$'' addresses the last message; thus the command **top** which prints the first few lines of a message could be used in ``top *'' to print the first few lines of all messages.

Replying to or originating mail. You can use the **reply** command to set up a response to a message, sending it back to the person who it was from. Text you then type in, up to an end-of-file (or a line consisting only of a ``.'') defines the contents of the message. While you are composing a message, *Mail* treats lines beginning with the character `~' specially. For instance, typing ``~m'' (alone on a line) will place a copy of the current message into the response right shifting it by a tabstop. Other escapes will set up subject fields, add and delete recipients to the message and allow you to escape to an editor to revise the message

or to a shell to run some commands. (These options will be given in the summary below.)

Ending a mail processing session. You can end a Mail session with the **quit** (**q**) command. Messages which have been examined go to your mbox file unless they have been deleted in which case they are discarded. Unexamined messages go back to the post office. The **-f** option causes Mail to read in the contents of your mbox (or the specified file) for processing; when you **quit** Mail writes undeleted messages back to this file.

Personal and systemwide distribution lists. It is also possible to create a personal distribution lists so that, for instance, you can send mail to ``cohorts'' and have it go to a group of people. Such lists can be defined by placing a line like

    alias cohorts bill ozalp sklower jkf mark cory!kridle

in the file .mailrc in your home directory. The current list of such aliases can be displayed by the **alias (a)** command in Mail. On systems running delivermail(8), system wide distribution lists can be created by editing /usr/lib/aliases, see aliases(5) and delivermail(8); these are kept in a slightly different syntax. In mail you send, personal aliases will be expanded in mail sent to others so that they will be able to **reply** to the recipients. System wide aliases are not expanded when the mail is sent, but any reply returned to the machine will have the system wide alias expanded as all mail goes through delivermail. If you edit /usr/lib/aliases, you must run the program newaliases(1).

Network mail (ARPA, UUCP, Berknet) Mail to sites on the ARPA network and sites within Bell laboratories can be sent using ``name@site'' for ARPA-net sites or ``machine!user'' for Bell labs sites, provided appropriate gateways are known to the system. (Be sure to escape the ! in Bell sites when giving it on a csh command line by preceding it with an \. Machines on an instance of the Berkeley network are addressed as ``machine:user'', e.g. ``csvax:bill''. When addressed from the arpa-net, ``csvax:bill'' is known as ``csvax.bill@berkeley''.

Mail has a number of options which can be **set** in the .mailrc file to alter its behavior; thus ``set askcc'' enables the ``askcc'' feature. (These options are summarized below.)

SUMMARY

(Adapted from the `Mail Reference Manual') Each command is typed on a line by itself, and may take arguments following the command word. The command need not be typed in its entirety - the first command which matches the typed prefix is used. For the commands which take message lists as arguments, if no message list is given, then the next message forward which satisfies the command's requirements is used. If there are no messages forward

of the current message, the search proceeds backwards, and if there are no good messages at all, _Mail_ types ``No applicable messages'' and aborts the command.

-	Goes to the previous message and prints it out. If given a numeric argument _n_ , goes to the _n th_ previous message and prints it.
?	Prints a brief summary of commands.
!	Executes the UNIX shell command which follows.
alias	(**a**) With no arguments, prints out all currently-defined aliases. With one argument, prints out that alias. With more than one argument, adds the users named in the second and later arguments to the alias named in the first argument.
chdir	(**c**) Changes the user's working directory to that specified, if given. If no directory is given, then changes to the user's login directory.
delete	(**d**) Takes a list of messages as argument and marks them all as deleted. Deleted messages will not be saved in _mbox_ , nor will they be available for most other commands.
dp	(also **dt**) Deletes the current message and prints the next message. If there is no next message, _Mail_ says ``at EOF.''
edit	(**e**) Takes a list of messages and points the text editor at each one in turn. On return from the editor, the message is read back in.
exit	(**ex** or **x**) Effects an immediate return to the Shell without modifying the user's system mailbox, his _mbox_ file, or his edit file in **-f** .
from	(**f**) Takes a list of messages and prints their message headers.
headers	(**h**) Lists the current range of headers, which is an 18 message group. If a ``+'' argument is given, then the next 18 message group is printed, and if a ``-'' argument is given, the previous 18 message group is printed.
help	A synonym for ?
hold	(**ho**, also **preserve**) Takes a message list and marks each message therein to be saved in the user's system

mailbox instead of in mbox. Does not override the **delete** command.

mail	(m) Takes as argument login names and distribution group names and sends mail to those people.
next	(n like + or CR) Goes to the next message in sequence and types it. With an argument list, types the next matching message.
preserve	A synonym for **hold**.
print	(p) Takes a message list and types out each message on the user's terminal.
quit	(q) Terminates the session, saving all undeleted, unsaved messages in the user's mbox file in his login directory, preserving all messages marked with **hold** or **preserve** or never referenced in his system mail-box, and removing all other messages from his system mailbox. If new mail has arrived during the session, the message ``You have new mail'' is given. If given while editing a mailbox file with the **-f** flag, then the edit file is rewritten. A return to the Shell is effected, unless the rewrite of edit file fails, in which case the user can escape with the **exit** command.
reply	(r) Takes a message list and sends mail to each message author and recipient just like the **mail** command. The default message must not be deleted.
Reply	(R) Just like **reply** except it only replies to the sender, not to any of the people who also received the message.
respond	A synonym for **reply** .
save	(s) Takes a message list and a filename and appends each message in turn to the end of the file. The filename in quotes, followed by the line count and character count is echoed on the user's terminal.
set	(se) With no arguments, prints all variable values. Otherwise, sets option. Arguments are of the form ``option=value'' or ``option.''
shell	(sh) Invokes an interactive version of the shell.
size	Takes a message list and prints out the size in characters of each message.
top	Takes a message list and prints the top few lines of

each.
The number of lines printed is controlled by the variable **toplines** and defaults to five.

type (t) A synonym for **print** .

unalias Takes a list of names defined by **alias** commands and discards the remembered groups of users. The group names no longer have any significance.

undelete (u) Takes a message list and marks each one as not being deleted.

unset Takes a list of option names and discards their remembered values; the inverse of **set** .

visual (v) Takes a message list and invokes the display editor on each message.

write (w) A synonym for **save** .

xit (x) A synonym for **exit** .

Here is a summary of the tilde escapes, which are used when composing messages to perform special functions. Tilde escapes are only recognized at the beginning of lines. The name ``tilde escape'' is somewhat of a misnomer since the actual escape character can be set by the option **escape**.

~!command Execute the indicated shell command, then return to the message.

~b name ... Add the given names to the list of blind carbon copy recipients.

~c name ... Add the given names to the list of carbon copy recipients.

~d Read the file ``dead.letter'' from your home directory into the message.

~e Invoke the text editor on the message collected so far. After the editing session is finished, you may continue appending text to the message.

~f messages Just like ~m except that the messages are not shifted to the right. This is well suited to forwarding mail to someone else.

~h Edit the message header fields by typing each one in turn and allowing the user to append text to the end or modify the field by using the current terminal

erase and kill characters.

~m messages Read the named messages into the message being sent, shifted right one tab. If no messages are specified, read the current message.

~p Print out the message collected so far, prefaced by the message header fields.

~q Abort the message being sent, copying the message to ``dead.letter'' in your home directory if **save** is set.

~r filename Read the named file into the message.

~s string Cause the named string to become the current subject field.

~t name ... Add the given names to the direct recipient list.

~v Invoke an alternate editor (defined by the VISUAL option) on the message collected so far. Usually, the alternate editor will be a screen editor. After you quit the editor, you may resume appending text to the end of your message.

~w filename Write the message onto the named file.

~|command Pipe the message through the command as a filter. If the command gives no output or terminates abnormally, retain the original text of the message. The command *fmt*(1) is often used as *command* to rejustify the message.

~~string Insert the string of text in the message prefaced by a single ~. If you have changed the escape character, then you should double that character in order to send it.

Options are controlled via the **set** and **unset** commands. Options may be either binary, in which case it is only significant to see whether they are set or not, or string, in which case the actual value is of interest. The binary options include the following:

append Causes messages saved in *mbox* to be appended to the end rather than prepended. (This is set in /usr/lib/Mail.rc on version 7 systems.)

ask Causes *Mail* to prompt you for the subject of each message you send. If you respond with simply a newline, no subject field will be sent.

askcc	Causes you to be prompted for additional carbon copy recipients at the end of each message. Responding with a newline indicates your satisfaction with the current list.
autoprint	Causes the **delete** command to behave like **dp** -- thus, after deleting a message, the next one will be typed automatically.
ignore	Causes interrupt signals from your terminal to be ignored and echoed as @'s.
metoo	Usually, when a group is expanded that contains the sender, the sender is removed from the expansion. Setting this option causes the sender to be included in the group.
quiet	Suppresses the printing of the version when first invoked.
save	Causes the message collected prior to a interrupt to be saved on the file ``dead.letter'' in your home directory on receipt of two interrupts (or after a **~q**.)

The following options have string values:

EDITOR	Pathname of the text editor to use in the **edit** command and ~e escape. If not defined, then a default editor is used.
SHELL	Pathname of the shell to use in the ! command and the ~! escape. A default shell is used if this option is not defined.
VISUAL	Pathname of the text editor to use in the **visual** command and ~v escape.
escape	If defined, the first character of this option gives the character to use in the place of ~ to denote escapes.
record	If defined, gives the pathname of the file used to record all outgoing mail. If not defined, then outgoing mail is not so saved.
toplines	If defined, gives the number of lines of a message to be printed out with the **top** command; normally, the first five lines are printed.

FILES

/usr/spool/mail/*   post office

~/mbox	your old mail
~/.mailrc	file giving initial mail commands
/tmp/R#	temporary for editor escape
/usr/lib/Mail.help*	help files
/usr/lib/Mail.rc	system initialization file
/bin/mail	to do actual mailing
/etc/delivermail	postman

SEE ALSO

binmail(1), fmt(1), newaliases(1), aliases(5), delivermail(8)
'The Mail Reference Manual'

AUTHOR

Kurt Shoens

BUGS

NAME

make − maintain program groups

SYNOPSIS

make [**−f** makefile] [**−p**] [**−i**] [**−k**] [**−s**] [**−r**] [**−n**] [**−b**] [**−e**] [**−m**] [**−t**] [**−q**] [**−d**] [name] ...

DESCRIPTION

Make executes commands in *makefile* to update one or more target *names. Name* is typically a program. If no **−f** option is present, **makefile, Makefile, s.makefile,** and **s.Makefile** are tried in order. If *makefile* is **−**, the standard input is taken. More than one **−f** makefile argument pair may appear.

Make updates a target only if it depends on files that are newer than the target. All prerequisite files of a target are added recursively to the list of targets. Missing files are deemed to be out of date.

Makefile contains a sequence of entries that specify dependencies. The first line of an entry is a blank-separated, non-null list of targets, then a colon, then a (possibly null) list of prerequisite files or dependencies. Text following a semicolon, and all following lines that begin with a tab, are Shell commands to be executed to update the target. The first line that does not begin with a tab or sharp begins a new dependency or macro definition. Shell commands may be continued across lines with the <backslash><newline> sequence. Sharp and new-line surround comments.

The following makefile says that 'pgm' depends on two files 'a.o' and 'b.o', and that they in turn depend on '.c' files and a common file 'incl.h'.

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

Command lines are executed one at a time, each by its own Shell. A line is printed when it is executed unless the **−s** option is present, or the entry **.SILENT:** is in *makefile*, or unless the first character of the command is @. The **−n** option specifies printing without execution; however, if the command line has the string "$(MAKE)" in it the line is always executed (see discussion of **MAKEFLAGS** macro below under "Environment"). The **−t** (touch) option updates the modified date of a file without executing any commands.

Commands returning nonzero status normally terminate *make.* If the **−i** option is present, or the entry **.IGNORE:** appears in *makefile*, or if the line specifying the command begins with <tab><hyphen>, the error is ignored. If the **−k** option is present, work is abandoned on the current entry, but continues on other branches that do not depend on that entry.

The **−b** option allows old makefiles (those written for the old version of make) to run without errors. The difference between the old version of make and this version is that the new version requires all dependency lines to have a (possibly null) command associated with them. The previous version of make assumed if no command was specified explicitly that the command was null. This was contrary to the documentation.

Interrupt and quit cause the target to be deleted unless the target depends on the special name **.PRECIOUS.**

The Environment

The environment is read by make. All variables are assumed to be macro definitions and processed as such. The environment variables are processed before any makefile and after the internal rules. Thus, macro assignments in a makefile override environment variables. The **−e** option causes the environment to override the macro assignments in a makefile.

The **MAKEFLAGS** environment variable is processed by make as containing any legal input option (except **-f**, **-p**, and **-d**) defined for the command line. Further, upon invocation, make "invents" the variable, if it is not in the environment, puts the current options into it, and passes it on to invocations of commands. Thus, **MAKEFLAGS** always contains the current input options. This proves very useful for "supermakes". In fact, as noted above, when the **-n** option is used, the command "$(MAKE)" is executed anyway; hence, one can perform a *make -n* recursively on a whole software system to see what would have been executed. This is because the **-n** is put in **MAKEFLAGS** and passed to further invocations of "$(MAKE)". This is one way of debugging all of the makefiles for a software project without actually doing anything.

Macros

Entries of the form

string1 = string2

are macro definitions. Subsequent appearances of *$(string1[:subst1=[subst2]])* are replaced by *string2*. (The parentheses are optional if a single character macro name is used and there is no substitute sequence.) The optional ":subst1=subst2" is an substitute sequence. If it is specified, all non-overlapping occurrences of "subst1" in the named macro are replaced by "subst2". "Strings" (for the purposes of this type of substitution) are delimited by blanks, tabs, newline character and the beginning of a line. An example of the use of the substitute sequence is given under the Libraries heading.

Internal Macros

There are five internally maintained macros which are useful for writing rules for building targets.

$*** The macro **$* stands for the file name part with the suffix deleted, of the current dependent. It is evaluated only for inference rules.

$@** The **$@ macro stands for the full target name of the current target. It is evaluated only for explicitly named dependencies.

$<** The **$< macro is only evaluated for inference rules or the **.DEFAULT** rule. It is the "thing" which is out of date with respect to the target (i.e. the "manufactured" dependent file name). Thus, in the ".c.o" rule the **$<** macro would evaluate to the ".c" file.

$?** The **$? macro is evaluated when explicit rules from the makefile are evaluated. It is the list of prerequisites that are out of date with respect to the target. (Essentially, those "things" which must be rebuilt.)

$% The **$%** macro is only evaluated when the target is an archive library member of the form "lib(file.o)". In this case, **$@** evaluates to "lib" and **$%** evaluates to the library member, "file.o".

An example: a rule for making optimized '.o' files from '.c' files is:

```
        .c.o:
                cc −c −O $*.c
```

or

```
        .c.o:
                cc −c −O $<
```

Four of the five macros can have alternative forms. When an upper case 'D' or 'F' is appended to any of the four macros the meaning is changed to "directory part" for 'D' and "file part" for