MAIL REFERENCE MANUAL

Kurt Shoens

Revised by

Craig Leres and *Mark Andrews*

Version 5.5

May 27, 1992

1. Introduction

Mail provides a simple and friendly environment for sending and receiving mail. It divides incoming mail into its constituent messages and allows the user to deal with them in any order. In addition, it provides a set of *ed*-like commands for manipulating messages and sending mail. *Mail* offers the user simple editing capabilities to ease the composition of outgoing messages, as well as providing the ability to define and send to names which address groups of users. Finally, *Mail* is able to send and receive messages across such networks as the ARPANET, UUCP, and Berkeley network.

This document describes how to use the *Mail* program to send and receive messages. The reader is not assumed to be familiar with other message handling systems, but should be familiar with the UNIX[1] shell, the text editor, and some of the common UNIX commands. ''The UNIX Programmer's Manual,'' ''An Introduction to Csh,'' and ''Text Editing with Ex and Vi'' can be consulted for more information on these topics.

Here is how messages are handled: the mail system accepts incoming *messages* for you from other people and collects them in a file, called your *system mailbox*. When you login, the system notifies you if there are any messages waiting in your system mailbox. If you are a *csh* user, you will be notified when new mail arrives if you inform the shell of the location of your mailbox. On version 7 systems, your system mailbox is located in the directory /usr/spool/mail in a file with your login name. If your login name is ''sam,'' then you can make *csh* notify you of new mail by including the following line in your .cshrc file:

    set mail=/usr/spool/mail/sam

When you read your mail using *Mail*, it reads your system mailbox and separates that file into the individual messages that have been sent to you. You can then read, reply to, delete, or save these messages. Each message is marked with its author and the date they sent it.

---

[1] UNIX is a trademark of Bell Laboratories.

2. Common usage

The *Mail* command has two distinct usages, according to whether one wants to send or receive mail. Sending mail is simple: to send a message to a user whose login name is, say, ‘‘root,’’ use the shell command:

% Mail root

then type your message. When you reach the end of the message, type an EOT (control−d) at the beginning of a line, which will cause *Mail* to echo ‘‘EOT’’ and return you to the Shell. When the user you sent mail to next logs in, he will receive the message:

You have mail.

to alert him to the existence of your message.

If, while you are composing the message you decide that you do not wish to send it after all, you can abort the letter with a RUBOUT. Typing a single RUBOUT causes *Mail* to print

(Interrupt -- one more to kill letter)

Typing a second RUBOUT causes *Mail* to save your partial letter on the file ‘‘dead.letter’’ in your home directory and abort the letter. Once you have sent mail to someone, there is no way to undo the act, so be careful.

The message your recipient reads will consist of the message you typed, preceded by a line telling who sent the message (your login name) and the date and time it was sent.

If you want to send the same message to several other people, you can list their login names on the command line. Thus,

```
% Mail sam bob john
Tuition fees are due next Friday.  Don't forget!!
<Control−d>
EOT
%
```

will send the reminder to sam, bob, and john.

If, when you log in, you see the message,

You have mail.

you can read the mail by typing simply:

% Mail

Mail will respond by typing its version number and date and then listing the messages you have waiting. Then it will type a prompt and await your command. The messages are assigned numbers starting with 1 − you refer to the messages with these numbers. *Mail* keeps track of which messages are *new* (have been sent since you last read your mail) and *read* (have been read by you). New messages have an **N** next to them in the header listing and old, but unread messages have a **U** next to them. *Mail* keeps track of new/old and read/unread messages by putting a header field called ‘‘Status’’ into your messages.

To look at a specific message, use the **type** command, which may be abbreviated to simply **t**. For example, if you had the following messages:

```
N 1 root     Wed Sep 21 09:21  "Tuition fees"
N 2 sam      Tue Sep 20 22:55
```

you could examine the first message by giving the command:

type 1

which might cause *Mail* to respond with, for example:

```
Message  1:
From root  Wed Sep 21 09:21:45 1978
Subject: Tuition fees
Status: R
```

Tuition fees are due next Wednesday. Don't forget!!

Many *Mail* commands that operate on messages take a message number as an argument like the **type** command. For these commands, there is a notion of a current message. When you enter the *Mail* program, the current message is initially the first one. Thus, you can often omit the message number and use, for example,

t

to type the current message. As a further shorthand, you can type a message by simply giving its message number. Hence,

1

would type the first message.

Frequently, it is useful to read the messages in your mailbox in order, one after another. You can read the next message in *Mail* by simply typing a newline. As a special case, you can type a newline as your first command to *Mail* to type the first message.

If, after typing a message, you wish to immediately send a reply, you can do so with the **reply** command. **Reply**, like **type**, takes a message number as an argument. *Mail* then begins a message addressed to the user who sent you the message. You may then type in your letter in reply, followed by a <control-d> at the beginning of a line, as before. *Mail* will type EOT, then type the ampersand prompt to indicate its readiness to accept another command. In our example, if, after typing the first message, you wished to reply to it, you might give the command:

reply

Mail responds by typing:

To: root
Subject: Re: Tuition fees

and waiting for you to enter your letter. You are now in the message collection mode described at the beginning of this section and *Mail* will gather up your message up to a control−d. Note that it copies the subject header from the original message. This is useful in that correspondence about a particular matter will tend to retain the same subject heading, making it easy to recognize. If there are other header fields in the message, the information found will also be used. For example, if the letter had a ''To:'' header listing several recipients, *Mail* would arrange to send your replay to the same people as well. Similarly, if the original message contained a ''Cc:'' (carbon copies to) field, *Mail* would send your reply to *those* users, too. *Mail* is careful, though, not too send the message to *you*, even if you appear in the ''To:'' or ''Cc:'' field, unless you ask to be included explicitly. See section 4 for more details.

After typing in your letter, the dialog with *Mail* might look like the following:

```
reply
To: root
Subject: Tuition fees

Thanks for the reminder
EOT
&
```

The **reply** command is especially useful for sustaining extended conversations over the message system, with other ''listening'' users receiving copies of the conversation. The **reply** command can be abbreviated to **r**.

Sometimes you will receive a message that has been sent to several people and wish to reply *only* to the person who sent it. **Reply** with a capital **R** replies to a message, but sends a copy to the sender only.

If you wish, while reading your mail, to send a message to someone, but not as a reply to one of your messages, you can send the message directly with the **mail** command, which takes as arguments the names of the recipients you wish to send to. For example, to send a message to ''frank,'' you would do:

mail frank

This is to confirm our meeting next Friday at 4.
EOT
&

The **mail** command can be abbreviated to **m**.

Normally, each message you receive is saved in the file *mbox* in your login directory at the time you leave *Mail*. Often, however, you will not want to save a particular message you have received because it is only of passing interest. To avoid saving a message in *mbox* you can delete it using the **delete** command. In our example,

delete 1

will prevent *Mail* from saving message 1 (from root) in *mbox*. In addition to not saving deleted messages, *Mail* will not let you type them, either. The effect is to make the message disappear altogether, along with its number. The **delete** command can be abbreviated to simply **d**.

Many features of *Mail* can be tailored to your liking with the **set** command. The **set** command has two forms, depending on whether you are setting a *binary* option or a *valued* option. Binary options are either on or off. For example, the ‘‘ask’’ option informs *Mail* that each time you send a message, you want it to prompt you for a subject header, to be included in the message. To set the ‘‘ask’’ option, you would type

set ask

Another useful *Mail* option is ‘‘hold.’’ Unless told otherwise, *Mail* moves the messages from your system mailbox to the file *mbox* in your home directory when you leave *Mail*. If you want *Mail* to keep your letters in the system mailbox instead, you can set the ‘‘hold’’ option.

Valued options are values which *Mail* uses to adapt to your tastes. For example, the ‘‘SHELL’’ option tells *Mail* which shell you like to use, and is specified by

set SHELL=/bin/csh

for example. Note that no spaces are allowed in ‘‘SHELL=/bin/csh.’’ A complete list of the *Mail* options appears in section 5.

Another important valued option is ‘‘crt.’’ If you use a fast video terminal, you will find that when you print long messages, they fly by too quickly for you to read them. With the ‘‘crt’’ option, you can make *Mail* print any message larger than a given number of lines by sending it through a paging program. This program is specified by the valued option **PAGER**. If **PAGER** is not set, a default paginator is used. For example, most CRT users with 24-line screens should do:

set crt=24

to paginate messages that will not fit on their screens. In the default state, *more* (default paginator) prints a screenful of information, then types --More--. Type a space to see the next screenful.

Another adaptation to user needs that *Mail* provides is that of *aliases*. An alias is simply a name which stands for one or more real user names. *Mail* sent to an alias is really sent to the list of real users associated with it. For example, an alias can be defined for the members of a project, so that you can send mail to the whole project by sending mail to just a single name. The **alias** command in *Mail* defines an alias. Suppose that the users in a project are named Sam, Sally, Steve, and Susan. To define an alias called ‘‘project’’ for them, you would use the *Mail* command:

alias project sam sally steve susan

The **alias** command can also be used to provide a convenient name for someone whose user name is inconvenient. For example, if a user named ‘‘Bob Anderson’’ had the login name ‘‘anderson,"’’ you might want to use:

alias bob anderson

so that you could send mail to the shorter name, ‘‘bob.’’

While the **alias** and **set** commands allow you to customize *Mail*, they have the drawback that they must be retyped each time you enter *Mail*. To make them more convenient to use, *Mail* always looks for two files when it is invoked. It first reads a system wide file ‘‘/usr/lib/Mail.rc,’’ then a user specific file,

‘‘.mailrc,’’ which is found in the user’s home directory. The system wide file is maintained by the system administrator and contains **set** commands that are applicable to all users of the system. The ‘‘.mailrc’’ file is usually used by each user to set options the way he likes and define individual aliases. For example, my .mailrc file looks like this:

set ask nosave SHELL=/bin/csh

As you can see, it is possible to set many options in the same **set** command. The ‘‘nosave’’ option is described in section 5.

Mail aliasing is implemented at the system-wide level by the mail delivery system *sendmail*. These aliases are stored in the file /usr/lib/aliases and are accessible to all users of the system. The lines in /usr/lib/aliases are of the form:

alias: $name_1$, $name_2$, $name_3$

where *alias* is the mailing list name and the $name_i$ are the members of the list. Long lists can be continued onto the next line by starting the next line with a space or tab. Remember that you must execute the shell command *newaliases* after editing /usr/lib/aliases since the delivery system uses an indexed file created by *newaliases*.

We have seen that *Mail* can be invoked with command line arguments which are people to send the message to, or with no arguments to read mail. Specifying the **–f** flag on the command line causes *Mail* to read messages from a file other than your system mailbox. For example, if you have a collection of messages in the file ‘‘letters’’ you can use *Mail* to read them with:

% Mail –f letters

You can use all the *Mail* commands described in this document to examine, modify, or delete messages from your ‘‘letters’’ file, which will be rewritten when you leave *Mail* with the **quit** command described below.

Since mail that you read is saved in the file *mbox* in your home directory by default, you can read *mbox* in your home directory by using simply

% Mail –f

Normally, messages that you examine using the **type** command are saved in the file ‘‘mbox’’ in your home directory if you leave *Mail* with the **quit** command described below. If you wish to retain a message in your system mailbox you can use the **preserve** command to tell *Mail* to leave it there. The **preserve** command accepts a list of message numbers, just like **type** and may be abbreviated to **pre**.

Messages in your system mailbox that you do not examine are normally retained in your system mailbox automatically. If you wish to have such a message saved in *mbox* without reading it, you may use the **mbox** command to have them so saved. For example,

mbox 2

in our example would cause the second message (from sam) to be saved in *mbox* when the **quit** command is executed. **Mbox** is also the way to direct messages to your *mbox* file if you have set the ‘‘hold’’ option described above. **Mbox** can be abbreviated to **mb**.

When you have perused all the messages of interest, you can leave *Mail* with the **quit** command, which saves the messages you have typed but not deleted in the file *mbox* in your login directory. Deleted messages are discarded irretrievably, and messages left untouched are preserved in your system mailbox so that you will see them the next time you type:

% Mail

The **quit** command can be abbreviated to simply **q**.

If you wish for some reason to leave *Mail* quickly without altering either your system mailbox or *mbox*, you can type the **x** command (short for **exit**), which will immediately return you to the Shell without changing anything.

If, instead, you want to execute a Shell command without leaving *Mail*, you can type the command preceded by an exclamation point, just as in the text editor. Thus, for instance:

    !date

will print the current date without leaving *Mail*.

Finally, the **help** command is available to print out a brief summary of the *Mail* commands, using only the single character command abbreviations.

3. Maintaining folders

Mail includes a simple facility for maintaining groups of messages together in folders. This section describes this facility.

To use the folder facility, you must tell *Mail* where you wish to keep your folders. Each folder of messages will be a single file. For convenience, all of your folders are kept in a single directory of your choosing. To tell *Mail* where your folder directory is, put a line of the form

    set folder=letters

in your *.mailrc* file. If, as in the example above, your folder directory does not begin with a '/,' *Mail* will assume that your folder directory is to be found starting from your home directory. Thus, if your home directory is **/usr/person** the above example told *Mail* to find your folder directory in **/usr/person/letters**.

Anywhere a file name is expected, you can use a folder name, preceded with '+.' For example, to put a message into a folder with the **save** command, you can use:

    save +classwork

to save the current message in the *classwork* folder. If the *classwork* folder does not yet exist, it will be created. Note that messages which are saved with the **save** command are automatically removed from your system mailbox.

In order to make a copy of a message in a folder without causing that message to be removed from your system mailbox, use the **copy** command, which is identical in all other respects to the **save** command. For example,

    copy +classwork

copies the current message into the *classwork* folder and leaves a copy in your system mailbox.

The **folder** command can be used to direct *Mail* to the contents of a different folder. For example,

    folder +classwork

directs *Mail* to read the contents of the *classwork* folder. All of the commands that you can use on your system mailbox are also applicable to folders, including **type**, **delete**, and **reply**. To inquire which folder you are currently editing, use simply:

    folder

To list your current set of folders, use the **folders** command.

To start *Mail* reading one of your folders, you can use the **–f** option described in section 2. For example:

    % Mail –f +classwork

will cause *Mail* to read your *classwork* folder without looking at your system mailbox.

4. More about sending mail

4.1. Tilde escapes

While typing in a message to be sent to others, it is often useful to be able to invoke the text editor on the partial message, print the message, execute a shell command, or do some other auxiliary function. *Mail* provides these capabilities through *tilde escapes*, which consist of a tilde (˜) at the beginning of a line, followed by a single character which indicates the function to be performed. For example, to print the text of the message so far, use:

˜p

which will print a line of dashes, the recipients of your message, and the text of the message so far. Since *Mail* requires two consecutive RUBOUT's to abort a letter, you can use a single RUBOUT to abort the output of ˜p or any other ˜ escape without killing your letter.

If you are dissatisfied with the message as it stands, you can invoke the text editor on it using the escape

˜e

which causes the message to be copied into a temporary file and an instance of the editor to be spawned. After modifying the message to your satisfaction, write it out and quit the editor. *Mail* will respond by typing

(continue)

after which you may continue typing text which will be appended to your message, or type <control-d> to end the message. A standard text editor is provided by *Mail*. You can override this default by setting the valued option ''EDITOR'' to something else. For example, you might prefer:

set EDITOR=/usr/ucb/ex

Many systems offer a screen editor as an alternative to the standard text editor, such as the *vi* editor from UC Berkeley. To use the screen, or *visual* editor, on your current message, you can use the escape,

˜v

˜v works like ˜e, except that the screen editor is invoked instead. A default screen editor is defined by *Mail*. If it does not suit you, you can set the valued option ''VISUAL'' to the path name of a different editor.

It is often useful to be able to include the contents of some file in your message; the escape

˜r filename

is provided for this purpose, and causes the named file to be appended to your current message. *Mail* complains if the file doesn't exist or can't be read. If the read is successful, the number of lines and characters appended to your message is printed, after which you may continue appending text. The filename may contain shell metacharacters like * and ? which are expanded according to the conventions of your shell.

As a special case of ˜r, the escape

˜d

reads in the file ''dead.letter'' in your home directory. This is often useful since *Mail* copies the text of your message there when you abort a message with RUBOUT.

To save the current text of your message on a file you may use the

˜w filename

escape. *Mail* will print out the number of lines and characters written to the file, after which you may continue appending text to your message. Shell metacharacters may be used in the filename, as in ˜r and are expanded with the conventions of your shell.

If you are sending mail from within *Mail's* command mode you can read a message sent to you into the message you are constructing with the escape:

˜m 4

which will read message 4 into the current message, shifted right by one tab stop. You can name any non-deleted message, or list of messages. Messages can also be forwarded without shifting by a tab stop with ˜f. This is the usual way to forward a message.

If, in the process of composing a message, you decide to add additional people to the list of message recipients, you can do so with the escape

˜t name1 name2 ...

You may name as few or many additional recipients as you wish. Note that the users originally on the recipient list will still receive the message; you cannot remove someone from the recipient list with ˜t.

If you wish, you can associate a subject with your message by using the escape

˜s Arbitrary string of text

which replaces any previous subject with ‘‘Arbitrary string of text.’’ The subject, if given, is sent near the top of the message prefixed with ‘‘Subject:’’ You can see what the message will look like by using ˜p.

For political reasons, one occasionally prefers to list certain people as recipients of carbon copies of a message rather than direct recipients. The escape

˜c name1 name2 ...

adds the named people to the ‘‘Cc:’’ list, similar to ˜t. Again, you can execute ˜p to see what the message will look like.

The escape

˜b name1 name2 ...

adds the named people to the ‘‘Cc:’’ list, but does not make the names visible in the ‘‘Cc:’’ line ("blind" carbon copy).

The recipients of the message together constitute the ‘‘To:’’ field, the subject the ‘‘Subject:’’ field, and the carbon copies the ‘‘Cc:’’ field. If you wish to edit these in ways impossible with the ˜t, ˜s, ˜c and ˜b escapes, you can use the escape

˜h

which prints ‘‘To:’’ followed by the current list of recipients and leaves the cursor (or printhead) at the end of the line. If you type in ordinary characters, they are appended to the end of the current list of recipients. You can also use your erase character to erase back into the list of recipients, or your kill character to erase them altogether. Thus, for example, if your erase and kill characters are the standard (on printing terminals) # and @ symbols,

```
˜h
To: root kurt####bill
```

would change the initial recipients ‘‘root kurt’’ to ‘‘root bill.’’ When you type a newline, *Mail* advances to the ‘‘Subject:’’ field, where the same rules apply. Another newline brings you to the ‘‘Cc:’’ field, which may be edited in the same fashion. Another newline brings you to the ‘‘Bcc:’’ ("blind" carbon copy) field, which follows the same rules as the "Cc:" field. Another newline leaves you appending text to the end of your message. You can use ˜p to print the current text of the header fields and the body of the message.

To effect a temporary escape to the shell, the escape

˜!command

is used, which executes *command* and returns you to mailing mode without altering the text of your message. If you wish, instead, to filter the body of your message through a shell command, then you can use

˜|command

which pipes your message through the command and uses the output as the new text of your message. If the command produces no output, *Mail* assumes that something is amiss and retains the old version of your message. A frequently-used filter is the command *fmt*, designed to format outgoing mail.

To effect a temporary escape to *Mail* command mode instead, you can use the

˜:*Mail command*

escape. This is especially useful for retyping the message you are replying to, using, for example:

~:t

It is also useful for setting options and modifying aliases.

If you wish abort the current message, you can use the escape

~q

This will terminate the current message and return you to the shell (or *Mail* if you were using the **mail** command). If the **save** option is set, the message will be copied to the file ‘‘dead.letter’’ in your home directory.

If you wish (for some reason) to send a message that contains a line beginning with a tilde, you must double it. Thus, for example,

~~This line begins with a tilde.

sends the line

~This line begins with a tilde.

Finally, the escape

~?

prints out a brief summary of the available tilde escapes.

On some terminals (particularly ones with no lower case) tilde’s are difficult to type. *Mail* allows you to change the escape character with the ‘‘escape’’ option. For example, I set

set escape=]

and use a right bracket instead of a tilde. If I ever need to send a line beginning with right bracket, I double it, just as for ~. Changing the escape character removes the special meaning of ~.

4.2. Network access

This section describes how to send mail to people on other machines. Recall that sending to a plain login name sends mail to that person on your machine. If your machine is directly (or sometimes, even, indirectly) connected to the Arpanet, you can send messages to people on the Arpanet using a name of the form

name@host.domain

where *name* is the login name of the person you’re trying to reach, *host* is the name of the machine on the Arpanet, and *domain* is the higher-level scope within which the hostname is known, e.g. EDU (for educational institutions), COM (for commercial entities), GOV (for governmental agencies), ARPA for many other things, BITNET or CSNET for those networks.

If your recipient logs in on a machine connected to yours by UUCP (the Bell Laboratories supplied network that communicates over telephone lines), sending mail can be a bit more complicated. You must know the list of machines through which your message must travel to arrive at his site. So, if his machine is directly connected to yours, you can send mail to him using the syntax:

host!name

where, again, *host* is the name of the machine and *name* is the login name. If your message must go through an intermediary machine first, you must use the syntax:

intermediary!host!name

and so on. It is actually a feature of UUCP that the map of all the systems in the network is not known anywhere (except where people decide to write it down for convenience). Talk to your system administrator about good ways to get places; the *uuname* command will tell you systems whose names are recognized, but not which ones are frequently called or well-connected.

When you use the **reply** command to respond to a letter, there is a problem of figuring out the names of the users in the ‘‘To:’’ and ‘‘Cc:’’ lists *relative to the current machine*. If the original letter was sent to you by someone on the local machine, then this problem does not exist, but if the message came from a remote machine, the problem must be dealt with. *Mail* uses a heuristic to build the correct name for each user relative to the local machine. So, when you **reply** to remote mail, the names in the ‘‘To:’’ and ‘‘Cc:’’ lists may change somewhat.

4.3. Special recipients

As described previously, you can send mail to either user names or **alias** names. It is also possible to send messages directly to files or to programs, using special conventions. If a recipient name has a ‘/’ in it or begins with a ‘+’, it is assumed to be the path name of a file into which to send the message. If the file already exists, the message is appended to the end of the file. If you want to name a file in your current directory (ie, one for which a ‘/’ would not usually be needed) you can precede the name with ‘./’ So, to send mail to the file ‘‘memo’’ in the current directory, you can give the command:

```
% Mail ./memo
```

If the name begins with a ‘+,’ it is expanded into the full path name of the folder name in your folder directory. This ability to send mail to files can be used for a variety of purposes, such as maintaining a journal and keeping a record of mail sent to a certain group of users. The second example can be done automatically by including the full pathname of the record file in the **alias** command for the group. Using our previous **alias** example, you might give the command:

```
alias project sam sally steve susan /usr/project/mail_record
```

Then, all mail sent to "project" would be saved on the file ‘‘/usr/project/mail_record’’ as well as being sent to the members of the project. This file can be examined using *Mail −f*.

It is sometimes useful to send mail directly to a program, for example one might write a project billboard program and want to access it using *Mail*. To send messages to the billboard program, one can send mail to the special name ‘|billboard’ for example. *Mail* treats recipient names that begin with a ‘|’ as a program to send the mail to. An **alias** can be set up to reference a ‘|’ prefaced name if desired. *Caveats*: the shell treats ‘|’ specially, so it must be quoted on the command line. Also, the ‘| program’ must be presented as a single argument to mail. The safest course is to surround the entire name with double quotes. This also applies to usage in the **alias** command. For example, if we wanted to alias ‘rmsgs’ to ‘rmsgs −s’ we would need to say:

```
alias rmsgs "| rmsgs -s"
```

5. Additional features

This section describes some additional commands useful for reading your mail, setting options, and handling lists of messages.

5.1. Message lists

Several *Mail* commands accept a list of messages as an argument. Along with **type** and **delete**, described in section 2, there is the **from** command, which prints the message headers associated with the message list passed to it. The **from** command is particularly useful in conjunction with some of the message list features described below.

A *message list* consists of a list of message numbers, ranges, and names, separated by spaces or tabs. Message numbers may be either decimal numbers, which directly specify messages, or one of the special characters ‘‘↑’’ ‘‘.’’ or ‘‘$’’ to specify the first relevant, current, or last relevant message, respectively. *Relevant* here means, for most commands ‘‘not deleted’’ and ‘‘deleted’’ for the **undelete** command.

A range of messages consists of two message numbers (of the form described in the previous paragraph) separated by a dash. Thus, to print the first four messages, use

    type 1−4

and to print all the messages from the current message to the last message, use

    type .−$

A *name* is a user name. The user names given in the message list are collected together and each message selected by other means is checked to make sure it was sent by one of the named users. If the message consists entirely of user names, then every message sent by one of those users that is *relevant* (in the sense described earlier) is selected. Thus, to print every message sent to you by ‘‘root,’’ do

    type root

As a shorthand notation, you can specify simply ‘‘*’’ to get every *relevant* (same sense) message. Thus,

    type *

prints all undeleted messages,

    delete *

deletes all undeleted messages, and

    undelete *

undeletes all deleted messages.

You can search for the presence of a word in subject lines with **/**. For example, to print the headers of all messages that contain the word ‘‘PASCAL,’’ do:

    from /pascal

Note that subject searching ignores upper/lower case differences.

5.2. List of commands

This section describes all the *Mail* commands available when receiving mail.

− The − command goes to the previous message and prints it. The − command may be given a decimal number *n* as an argument, in which case the *n*th previous message is gone to and printed.

? Prints a brief summary of commands.

! Used to preface a command to be executed by the shell.

Print
Like **print**, but also print out ignored header fields. See also **print**, **ignore** and **retain**. **Print** can be abbreviated to **P**.

Reply or **Respond**

Note the capital **R** in the name. Frame a reply to a one or more messages. The reply (or replies if you are using this on multiple messages) will be sent ONLY to the person who sent you the message (respectively, the set of people who sent the messages you are replying to). You can add people using the **~t**, **~c** and **~b** tilde escapes. The subject in your reply is formed by prefacing the subject in the original message with ''Re:'' unless it already began thus. If the original message included a ''reply-to'' header field, the reply will go *only* to the recipient named by ''reply-to.'' You type in your message using the same conventions available to you through the **mail** command. The **Reply** command is especially useful for replying to messages that were sent to enormous distribution groups when you really just want to send a message to the originator. Use it often. **Reply** (and **Respond**) can be abbreviated to **R**.

Type

Identical to the **Print** command. **Type** can be abbreviated to **T**.

alias Define a name to stand for a set of other names. This is used when you want to send messages to a certain group of people and want to avoid retyping their names. For example

alias project john sue willie kathryn

creates an alias *project* which expands to the four people John, Sue, Willie, and Kathryn. If no arguments are given, all currently-defined aliases are printed. If one argument is given, that alias is printed (if it exists). **Alias** can be abbreviated to **a**.

alternates

If you have accounts on several machines, you may find it convenient to use the /usr/lib/aliases on all the machines except one to direct your mail to a single account. The **alternates** command is used to inform *Mail* that each of these other addresses is really *you*. *Alternates* takes a list of user names and remembers that they are all actually you. When you **reply** to messages that were sent to one of these alternate names, *Mail* will not bother to send a copy of the message to this other address (which would simply be directed back to you by the alias mechanism). If *alternates* is given no argument, it lists the current set of alternate names. **Alternates** is usually used in the .mailrc file. **Alternates** can be abbreviated to **alt**.

chdir

The **chdir** command allows you to change your current directory. **Chdir** takes a single argument, which is taken to be the pathname of the directory to change to. If no argument is given, **chdir** changes to your home directory. **Chdir** can be abbreviated to **c**.

copy

The **copy** command does the same thing that **save** does, except that it does not mark the messages it is used on for deletion when you quit. **Copy** can be abbreviated to **co**.

delete

Deletes a list of messages. Deleted messages can be reclaimed with the **undelete** command. **Delete** can be abbreviated to **d**.

dp or **dt**

These commands delete the current message and print the next message. They are useful for quickly reading and disposing of mail. If there is no next message, *mail* says ''at EOF.''

edit To edit individual messages using the text editor, the **edit** command is provided. The **edit** command takes a list of messages as described under the **type** command and processes each by writing it into the file Message*x* where *x* is the message number being edited and executing the text editor on it. When you have edited the message to your satisfaction, write the message out and quit, upon which *Mail* will read the message back and remove the file. **Edit** can be abbreviated to **e**.

else Marks the end of the then-part of an **if** statement and the beginning of the part to take effect if the condition of the **if** statement is false.

endif

Marks the end of an **if** statement.

exit or **xit**

Leave *Mail* without updating the system mailbox or the file your were reading. Thus, if you accidentally delete several messages, you can use **exit** to avoid scrambling your mailbox. **Exit** can be abbreviated to **ex** or **x**.

file The same as **folder**. **File** can be abbreviated to **fi**.

folders
List the names of the folders in your folder directory.

folder
The **folder** command switches to a new mail file or folder. With no arguments, it tells you which file you are currently reading. If you give it an argument, it will write out changes (such as deletions) you have made in the current file and read the new file. Some special conventions are recognized for the name:

Name	Meaning
#	Previous file read
%	Your system mailbox
%name	*Name*'s system mailbox
&	Your ˜/mbox file
+folder	A file in your folder directory

Folder can be abbreviated to **fo**.

from
The **from** command takes a list of messages and prints out the header lines for each one; hence

from joe

is the easy way to display all the message headers from ‘‘joe.’’ **From** can be abbreviated to **f**.

headers
When you start up *Mail* to read your mail, it lists the message headers that you have. These headers tell you who each message is from, when they were received, how many lines and characters each message is, and the ‘‘Subject:’’ header field of each message, if present. In addition, *Mail* tags the message header of each message that has been the object of the **preserve** command with a ‘‘P.’’ Messages that have been **saved** or **written** are flagged with a ‘‘*.’’ Finally, **deleted** messages are not printed at all. If you wish to reprint the current list of message headers, you can do so with the **headers** command. The **headers** command (and thus the initial header listing) only lists the first so many message headers. The number of headers listed depends on the speed of your terminal. This can be overridden by specifying the number of headers you want with the *window* option. *Mail* maintains a notion of the current ‘‘window’’ into your messages for the purposes of printing headers. Use the **z** command to move forward and back a window. You can move *Mail's* notion of the current window directly to a particular message by using, for example,

headers 40

to move *Mail's* attention to the messages around message 40. If a ‘‘+’’ argument is given, then the next screenful of message headers is printed, and if a ‘‘−’’ argument is given, the previous screenful of message headers is printed. **Headers** can be abbreviated to **h**.

help Print a brief and usually out of date help message about the commands in *Mail*. The *man* page for *mail* is usually more up-to-date than either the help message or this manual. It is also a synonym for **?**.

hold Arrange to hold a list of messages in the system mailbox, instead of moving them to the file *mbox* in your home directory. If you set the binary option *hold*, this will happen by default. It does not override the **delete** command. **Hold** can be abbreviated to **ho**.

if Commands in your ‘‘.mailrc’’ file can be executed conditionally depending on whether you are sending or receiving mail with the **if** command. For example, you can do:

```
if receive
        commands...
endif
```

An **else** form is also available:

if send
commands...
else
commands...
endif

Note that the only allowed conditions are **receive** and **send**.

ignore
N.B.: *Ignore* has been superseded by *retain.*
Add the list of header fields named to the *ignore list.* Header fields in the ignore list are not printed on your terminal when you print a message. This allows you to suppress printing of certain machine-generated header fields, such as *Via* which are not usually of interest. The **Type** and **Print** commands can be used to print a message in its entirety, including ignored fields. If **ignore** is executed with no arguments, it lists the current set of ignored fields.

list List the valid *Mail* commands. **List** can be abbreviated to **l**.

mail Send mail to one or more people. If you have the *ask* option set, *Mail* will prompt you for a subject to your message. Then you can type in your message, using tilde escapes as described in section 4 to edit, print, or modify your message. To signal your satisfaction with the message and send it, type control-d at the beginning of a line, or a . alone on a line if you set the option *dot*. To abort the message, type two interrupt characters (RUBOUT by default) in a row or use the **~q** escape. The **mail** command can be abbreviated to **m**.

mbox
Indicate that a list of messages be sent to *mbox* in your home directory when you quit. This is the default action for messages if you do *not* have the *hold* option set.

next or **+**
The **next** command goes to the next message and types it. If given a message list, **next** goes to the first such message and types it. Thus,

next root

goes to the next message sent by ‘‘root’’ and types it. The **next** command can be abbreviated to simply a newline, which means that one can go to and type a message by simply giving its message number or one of the magic characters ‘‘^’’ ‘‘.’’ or ‘‘$’’. Thus,

.

prints the current message and

4

prints message 4, as described previously. **Next** can be abbreviated to **n**.

preserve
Same as **hold**. Cause a list of messages to be held in your system mailbox when you quit. **Preserve** can be abbreviated to **pre**.

print
Print the specified messages. If the **crt** variable is set, messages longer than the number of lines it indicates are paged through the command specified by the **PAGER** variable. The **print** command can be abbreviated to **p**.

quit Terminates the session, saving all undeleted, unsaved and unwritten messages in the user’s *mbox* file in their login directory (messages marked as having been read), preserving all messages marked with **hold** or **preserve** or never referenced in their system mailbox. Any messages that were deleted, saved, written or saved to *mbox* are removed from their 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. **Quit** can be abbreviated to **q**.

reply or **respond**
Frame a reply to a single message. The reply will be sent to the person who sent you the message (to

which you are replying), plus all the people who received the original message, except you. You can add people using the **~t**, **~c** and **~b** tilde escapes. The subject in your reply is formed by prefacing the subject in the original message with ''Re:'' unless it already began thus. If the original message included a ''reply-to'' header field, the reply will go *only* to the recipient named by ''reply-to.'' You type in your message using the same conventions available to you through the **mail** command. The **reply** (and **respond**) command can be abbreviated to **r**.

retain
Add the list of header fields named to the *retained list*. Only the header fields in the retain list are shown on your terminal when you print a message. All other header fields are suppressed. The **Type** and **Print** commands can be used to print a message in its entirety. If **retain** is executed with no arguments, it lists the current set of retained fields.

save It is often useful to be able to save messages on related topics in a file. The **save** command gives you the ability to do this. The **save** command takes as an argument a list of message numbers, followed by the name of the file in which to save the messages. The messages are appended to the named file, thus allowing one to keep several messages in the file, stored in the order they were put there. The filename in quotes, followed by the line count and character count is echoed on the user's terminal. An example of the **save** command relative to our running example is:

    s 1 2 tuitionmail

Saved messages are not automatically saved in *mbox* at quit time, nor are they selected by the **next** command described above, unless explicitly specified. **Save** can be abbreviated to **s**.

set Set an option or give an option a value. Used to customize *Mail*. Section 5.3 contains a list of the options. Options can be *binary*, in which case they are *on* or *off*, or *valued*. To set a binary option *option on*, do

    set option

To give the valued option *option* the value *value*, do

    set option=value

There must be no space before or after the ''='' sign. If no arguments are given, all variable values are printed. Several options can be specified in a single **set** command. **Set** can be abbreviated to **se**.

shell The **shell** command allows you to escape to the shell. **Shell** invokes an interactive shell and allows you to type commands to it. When you leave the shell, you will return to *Mail*. The shell used is a default assumed by *Mail*; you can override this default by setting the valued option ''SHELL,'' eg:

    set SHELL=/bin/csh

Shell can be abbreviated to **sh**.

size Takes a message list and prints out the size in characters of each message.

source
The **source** command reads *mail* commands from a file. It is useful when you are trying to fix your ''.mailrc'' file and you need to re-read it. **Source** can be abbreviated to **so**.

top The **top** command takes a message list and prints the first five lines of each addressed message. If you wish, you can change the number of lines that **top** prints out by setting the valued option ''toplines.'' On a CRT terminal,

    set toplines=10

might be preferred. **Top** can be abbreviated to **to**.

type Same as **print**. Takes a message list and types out each message on the terminal. The **type** command can be abbreviated to **t**.

undelete
Takes a message list and marks each message as *not* being deleted. **Undelete** can be abbreviated to **u**.

unread
Takes a message list and marks each message as *not* having been read. **Unread** can be abbreviated

to **U**.

unset

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

visual

It is often useful to be able to invoke one of two editors, based on the type of terminal one is using. To invoke a display oriented editor, you can use the **visual** command. The operation of the **visual** command is otherwise identical to that of the **edit** command.

Both the **edit** and **visual** commands assume some default text editors. These default editors can be overridden by the valued options ‘‘EDITOR’’ and ‘‘VISUAL’’ for the standard and screen editors. You might want to do:

set EDITOR=/usr/ucb/ex VISUAL=/usr/ucb/vi

Visual can be abbreviated to **v**.

write

The **save** command always writes the entire message, including the headers, into the file. If you want to write just the message itself, you can use the **write** command. The **write** command has the same syntax as the **save** command, and can be abbreviated to simply **w**. Thus, we could write the second message by doing:

w 2 file.c

As suggested by this example, the **write** command is useful for such tasks as sending and receiving source program text over the message system. The filename in quotes, followed by the line count and character count is echoed on the user’s terminal.

z *Mail* presents message headers in windowfuls as described under the **headers** command. You can move *Mail’s* attention forward to the next window by giving the

z+

command. Analogously, you can move to the previous window with:

z−

5.3. Custom options

Throughout this manual, we have seen examples of binary and valued options. This section describes each of the options in alphabetical order, including some that you have not seen yet. To avoid confusion, please note that the options are either all lower case letters or all upper case letters. When I start a sentence such as: ‘‘Ask’’ causes *Mail* to prompt you for a subject header, I am only capitalizing ‘‘ask’’ as a courtesy to English.

EDITOR

The valued option ‘‘EDITOR’’ defines the pathname of the text editor to be used in the **edit** command and ˜e. If not defined, a standard editor is used.

PAGER

Pathname of the program to use for paginating output when it exceeds *crt* lines. A default paginator is used if this option is not defined.

SHELL

The valued option ‘‘SHELL’’ gives the path name of your shell. This shell is used for the **!** command and ˜! escape. In addition, this shell expands file names with shell metacharacters like * and ? in them.

VISUAL

The valued option ‘‘VISUAL’’ defines the pathname of the screen editor to be used in the **visual** command and ˜v escape. A standard screen editor is used if you do not define one.

append

The ‘‘append’’ option is binary and causes messages saved in *mbox* to be appended to the end rather than prepended. Normally, *Mail* will put messages in *mbox* in the same order that the system puts messages in your system mailbox. By setting ‘‘append,’’ you are requesting that *mbox* be appended to regardless. It is in any event quicker to append.

ask ‘‘Ask’’ is a binary option which 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
‘‘Askcc’’ is a binary option which causes you to be prompted for additional carbon copy recipients at the end of each message. Responding with a newline shows your satisfaction with the current list.

autoprint
‘‘Autoprint’’ is a binary option which causes the **delete** command to behave like **dp** – thus, after deleting a message, the next one will be typed automatically. This is useful when quickly scanning and deleting messages in your mailbox.

crt The valued option is used as a threshold to determine how long a message must be before **PAGER** is used to read it.

debug
The binary option ‘‘debug’’ causes debugging information to be displayed. Use of this option is the same as using the **−d** command line flag.

dot ‘‘Dot’’ is a binary option which, if set, causes *Mail* to interpret a period alone on a line as the terminator of the message you are sending.

escape
To allow you to change the escape character used when sending mail, you can set the valued option ‘‘escape.’’ Only the first character of the ‘‘escape’’ option is used, and it must be doubled if it is to appear as the first character of a line of your message. If you change your escape character, then ˜ loses all its special meaning, and need no longer be doubled at the beginning of a line.

folder
The name of the directory to use for storing folders of messages. If this name begins with a ‘/’ *Mail* considers it to be an absolute pathname; otherwise, the folder directory is found relative to your home directory.

hold The binary option ‘‘hold’’ causes messages that have been read but not manually dealt with to be held in the system mailbox. This prevents such messages from being automatically swept into your *mbox* file.

ignore
The binary option ‘‘ignore’’ causes RUBOUT characters from your terminal to be ignored and echoed as @’s while you are sending mail. RUBOUT characters retain their original meaning in *Mail* command mode. Setting the ‘‘ignore’’ option is equivalent to supplying the **−i** flag on the command line as described in section 6.

ignoreeof
An option related to ‘‘dot’’ is ‘‘ignoreeof’’ which makes *Mail* refuse to accept a control−d as the end of a message. ‘‘Ignoreeof’’ also applies to *Mail* command mode.

keep
The ‘‘keep’’ option causes *Mail* to truncate your system mailbox instead of deleting it when it is empty. This is useful if you elect to protect your mailbox, which you would do with the shell command:

chmod 600 /usr/spool/mail/yourname

where *yourname* is your login name. If you do not do this, anyone can probably read your mail, although people usually don’t.

keepsave
When you **save** a message, *Mail* usually discards it when you **quit**. To retain all saved messages, set the ‘‘keepsave’’ option.

metoo
When sending mail to an alias, *Mail* makes sure that if you are included in the alias, that mail will not be sent to you. This is useful if a single alias is being used by all members of the group. If

however, you wish to receive a copy of all the messages you send to the alias, you can set the binary option ‘‘metoo.’’

noheader

The binary option ‘‘noheader’’ suppresses the printing of the version and headers when *Mail* is first invoked. Setting this option is the same as using **−N** on the command line.

nosave

Normally, when you abort a message with two RUBOUTs, *Mail* copies the partial letter to the file ‘‘dead.letter’’ in your home directory. Setting the binary option ‘‘nosave’’ prevents this.

Replyall

Reverses the sense of *reply* and *Reply* commands.

quiet

The binary option ‘‘quiet’’ suppresses the printing of the version when *Mail* is first invoked, as well as printing the for example ‘‘Message 4:’’ from the **type** command.

record

If you love to keep records, then the valued option ‘‘record’’ can be set to the name of a file to save your outgoing mail. Each new message you send is appended to the end of the file.

screen

When *Mail* initially prints the message headers, it determines the number to print by looking at the speed of your terminal. The faster your terminal, the more it prints. The valued option ‘‘screen’’ overrides this calculation and specifies how many message headers you want printed. This number is also used for scrolling with the **z** command.

sendmail

To use an alternate mail delivery system, set the ‘‘sendmail’’ option to the full pathname of the program to use. Note: this is not for everyone! Most people should use the default delivery system.

toplines

The valued option ‘‘toplines’’ defines the number of lines that the ‘‘top’’ command will print out instead of the default five lines.

verbose

The binary option "verbose" causes *Mail* to invoke sendmail with the **−v** flag, which causes it to go into verbose mode and announce expansion of aliases, etc. Setting the "verbose" option is equivalent to invoking *Mail* with the **−v** flag as described in section 6.

6. Command line options

This section describes command line options for *Mail* and what they are used for.

−N Suppress the initial printing of headers.

−d Turn on debugging information. Not of general interest.

−f file
Show the messages in *file* instead of your system mailbox. If *file* is omitted, *Mail* reads *mbox* in your home directory.

−i Ignore tty interrupt signals. Useful on noisy phone lines, which generate spurious RUBOUT or DELETE characters. It's usually more effective to change your interrupt character to control−c, for which see the *stty* shell command.

−n Inhibit reading of /usr/lib/Mail.rc. Not generally useful, since /usr/lib/Mail.rc is usually empty.

−s string
Used for sending mail. *String* is used as the subject of the message being composed. If *string* contains blanks, you must surround it with quote marks.

−u name
Read *names's* mail instead of your own. Unwitting others often neglect to protect their mailboxes, but discretion is advised. Essentially, **−u user** is a shorthand way of doing **−f /usr/spool/mail/user**.

−v Use the −**v** flag when invoking sendmail. This feature may also be enabled by setting the the option "verbose".

The following command line flags are also recognized, but are intended for use by programs invoking *Mail* and not for people.

−T file
Arrange to print on *file* the contents of the *article-id* fields of all messages that were either read or deleted. **−T** is for the *readnews* program and should NOT be used for reading your mail.

−h number
Pass on hop count information. *Mail* will take the number, increment it, and pass it with −**h** to the mail delivery system. **−h** only has effect when sending mail and is used for network mail forwarding.

−r name
Used for network mail forwarding: interpret *name* as the sender of the message. The *name* and **−r** are simply sent along to the mail delivery system. Also, *Mail* will wait for the message to be sent and return the exit status. Also restricts formatting of message.

Note that **−h** and **−r**, which are for network mail forwarding, are not used in practice since mail forwarding is now handled separately. They may disappear soon.

7. Format of messages

This section describes the format of messages. Messages begin with a *from* line, which consists of the word ‘‘From’’ followed by a user name, followed by anything, followed by a date in the format returned by the *ctime* library routine described in section 3 of the Unix Programmer's Manual. A possible *ctime* format date is:

```
Tue Dec  1 10:58:23 1981
```

The *ctime* date may be optionally followed by a single space and a time zone indication, which should be three capital letters, such as PDT.

Following the *from* line are zero or more *header field* lines. Each header field line is of the form:

```
name: information
```

Name can be anything, but only certain header fields are recognized as having any meaning. The recognized header fields are: *article-id*, *bcc*, *cc*, *from*, *reply-to*, *sender*, *subject*, and *to*. Other header fields are also significant to other systems; see, for example, the current Arpanet message standard for much more information on this topic. A header field can be continued onto following lines by making the first character on the following line a space or tab character.

If any headers are present, they must be followed by a blank line. The part that follows is called the *body* of the message, and must be ASCII text, not containing null characters. Each line in the message body must be no longer than 512 characters and terminated with an ASCII newline character. If binary data must be passed through the mail system, it is suggested that this data be encoded in a system which encodes six bits into a printable character (i.e.: uuencode). For example, one could use the upper and lower case letters, the digits, and the characters comma and period to make up the 64 characters. Then, one can send a 16-bit binary number as three characters. These characters should be packed into lines, preferably lines about 70 characters long as long lines are transmitted more efficiently.

The message delivery system always adds a blank line to the end of each message. This blank line must not be deleted.

The UUCP message delivery system sometimes adds a blank line to the end of a message each time it is forwarded through a machine.

It should be noted that some network transport protocols enforce limits to the lengths of messages.

8. Glossary

This section contains the definitions of a few phrases peculiar to *Mail*.

alias An alternative name for a person or list of people.

flag An option, given on the command line of *Mail*, prefaced with a –. For example, **–f** is a flag.

header field
At the beginning of a message, a line which contains information that is part of the structure of the message. Popular header fields include *to*, *cc*, and *subject*.

mail A collection of messages. Often used in the phrase, ‘‘Have you read your mail?’’

mailbox
The place where your mail is stored, typically in the directory /usr/spool/mail.

message
A single letter from someone, initially stored in your *mailbox*.

message list
A string used in *Mail* command mode to describe a sequence of messages.

option
A piece of special purpose information used to tailor *Mail* to your taste. Options are specified with the **set** command.

9. Summary of commands, options, and escapes

This section gives a quick summary of the *Mail* commands, binary and valued options, and tilde escapes.

The following table describes the commands:

Command	*Description*
+	Same as **next**
-	Back up to previous message
?	Print brief summary of *Mail* commands
!	Single command escape to shell
Print	Type message with ignored fields
Reply	Reply to author of message only
Respond	Same as **Reply**
Type	Type message with ignored fields
alias	Define an alias as a set of user names
alternates	List other names you are known by
chdir	Change working directory, home by default
copy	Copy a message to a file or folder
delete	Delete a list of messages
dp	Same as **dt**
dt	Delete current message, type next message
edit	Edit a list of messages
else	Start of else part of conditional; see **if**
endif	End of conditional statement; see **if**
exit	Leave mail without changing anything
file	Interrogate/change current mail file
folder	Same as **file**
folders	List the folders in your folder directory
from	List headers of a list of messages
headers	List current window of messages
help	Same as **?**
hold	Same as **preserve**
if	Conditional execution of *Mail* commands
ignore	Set/examine list of ignored header fields
list	List valid *Mail* commands
local	List other names for the local host
mail	Send mail to specified names
mbox	Arrange to save a list of messages in *mbox*
next	Go to next message and type it
preserve	Arrange to leave list of messages in system mailbox
print	Print messages
quit	Leave *Mail*; update system mailbox, *mbox* as appropriate
reply	Compose a reply to a message
respond	Same as **reply**
retain	Supersedes **ignore**
save	Append messages, headers included, on a file
set	Set binary or valued options
shell	Invoke an interactive shell
size	Prints out size of message list
source	Read *mail* commands from a file
top	Print first so many (5 by default) lines of list of messages
type	Same as **print**
undelete	Undelete list of messages
unread	Marks list of messages as not been read
unset	Undo the operation of a **set**

visual	Invoke visual editor on a list of messages
write	Append messages to a file, don’t include headers
xit	Same as **exit**
z	Scroll to next/previous screenful of headers

The following table describes the options. Each option is shown as being either a binary or valued option.

Option	*Type*	*Description*
EDITOR	*valued*	Pathname of editor for ˜e and **edit**
PAGER	*valued*	Pathname of paginator for **Print**, **print**, **Type** and **type**
SHELL	*valued*	Pathname of shell for **shell**, ˜! and **!**
VISUAL	*valued*	Pathname of screen editor for ˜v, **visual**
append	*binary*	Always append messages to end of *mbox*
ask	*binary*	Prompt user for Subject: field when sending
askcc	*binary*	Prompt user for additional Cc's at end of message
autoprint	*binary*	Print next message after **delete**
crt	*valued*	Minimum number of lines before using **PAGER**
debug	*binary*	Print out debugging information
dot	*binary*	Accept . alone on line to terminate message input
escape	*valued*	Escape character to be used instead of ˜
folder	*valued*	Directory to store folders in
hold	*binary*	Hold messages in system mailbox by default
ignore	*binary*	Ignore RUBOUT while sending mail
ignoreeof	*binary*	Don't terminate letters/command input with ↑**D**
keep	*binary*	Don't unlink system mailbox when empty
keepsave	*binary*	Don't delete **save**d messages by default
metoo	*binary*	Include sending user in aliases
noheader	*binary*	Suppress initial printing of version and headers
nosave	*binary*	Don't save partial letter in *dead.letter*
quiet	*binary*	Suppress printing of *Mail* version and message numbers
record	*valued*	File to save all outgoing mail in
screen	*valued*	Size of window of message headers for **z**, etc.
sendmail	*valued*	Choose alternate mail delivery system
toplines	*valued*	Number of lines to print in **top**
verbose	*binary*	Invoke sendmail with the **−v** flag

The following table summarizes the tilde escapes available while sending mail.

Escape	*Arguments*	*Description*
˜!	*command*	Execute shell command
˜b	*name ...*	Add names to "blind" Cc: list
˜c	*name ...*	Add names to Cc: field
˜d		Read *dead.letter* into message
˜e		Invoke text editor on partial message
˜f	*messages*	Read named messages
˜h		Edit the header fields
˜m	*messages*	Read named messages, right shift by tab
˜p		Print message entered so far
˜q		Abort entry of letter; like RUBOUT
˜r	*filename*	Read file into message
˜s	*string*	Set Subject: field to *string*
˜t	*name ...*	Add names to To: field
˜v		Invoke screen editor on message
˜w	*filename*	Write message on file
˜|	*command*	Pipe message through *command*
˜:	*Mail command*	Execute a *Mail* command
˜˜	*string*	Quote a ˜ in front of *string*

The following table shows the command line flags that *Mail* accepts:

Flag	Description
−N	Suppress the initial printing of headers
−T *file*	Article-id's of read/deleted messages to *file*
−d	Turn on debugging
−f *file*	Show messages in *file* or *~/mbox*
−h *number*	Pass on hop count for mail forwarding
−i	Ignore tty interrupt signals
−n	Inhibit reading of /usr/lib/Mail.rc
−r *name*	Pass on *name* for mail forwarding
−s *string*	Use *string* as subject in outgoing mail
−u *name*	Read *name's* mail instead of your own
−v	Invoke sendmail with the **−v** flag

Notes: **−T**, **−d**, **−h**, and **−r** are not for human use.