IEEE P1003.2a Draft 8 – December 1991

`dot` When `dot` is set, a period on a line by itself during message input from a terminal shall also signify end-of-file (in addition to normal end-of-file). The default shall be `nodot`. If `ignoreeof` is set 7
(see below), a setting of `nodot` shall be ignored and the period 7
shall be the only method to terminate input mode. 7

`escape=`*c* Set the command escape character to be the character *c*. By default, the command escape character shall be tilde. If `escape` is unset, tilde shall be used; if it is set to null, command escaping shall be disabled.

`flipr` Reverse the meanings of the `R` and `r` commands; see 4.40.7.2.25 7
and 4.40.7.2.26. The default shall be `noflipr`. 7

`folder=`*directory*

The default directory for saving mail files. User-specified filenames beginning with a plus-sign (+) shall be expanded by preceding the filename with this directory name to obtain the real pathname. If *directory* does not start with a slash (/), the contents of **HOME** shall be prefixed to it. The default shall be `nofolder`. If `folder` is unset or set to null, user-specified filenames beginning with '`+`' shall refer to files in the current directory that begin with the literal `+` character. See also `outfolder` below. The `folder` value need not affect the processing of the files named in **MBOX** and **DEAD** .

`header` Enable writing of the header summary when entering `mailx` in Receive Mode. The default shall be `header`.

`hold` Preserve all messages that are read in the system mailbox instead of putting them in the *mbox* save file. The default shall be `nohold`.

`ignore` Ignore interrupts while entering messages. The default shall be `noignore`.

`ignoreeof` Ignore normal end-of-file during message input. Input can be ter- 7
minated only by entering a period (`.`) on a line by itself or by the 7
`~.` command escape. The default shall be `noignoreeof`. See also `dot` above.

`indentprefix=`*string*

A string that shall be prefixed to each line that is inserted into the message by the `~m` command escape. This variable shall default to one `<tab>` character.

`keep` When a system mailbox, secondary mailbox, or *mbox* is empty, truncate it to zero length instead of removing it. The default shall be `nokeep`.

`keepsave` Keep messages that have been saved in other files in the system mailbox instead of deleting them. The default shall be `nokeepsave`.

`metoo` Suppress the deletion of the login name of the user from the recipient list when replying to a message or sending to a group. The default shall be `nometoo`.

`outfolder` Cause the files used to record outgoing messages to be located in the directory specified by the `folder` variable unless the pathname is absolute. The default shall be `nooutfolder`. See the `record` variable.

`page` Insert a `<form-feed>` after each message sent through the pipe created by the `pipe` command. The default shall be `nopage`.

`prompt=`*string*
: Set the command-mode prompt to *string* . If *string* is null or if `noprompt` is set, no prompting shall occur. The default shall be to prompt with the string `"? "`.

`quiet` Refrain from writing the opening message and version when entering `mailx`. The default shall be `noquiet`.

`record=`*file*
: Record all outgoing mail in the file with the pathname *file* . The default shall be `norecord`. See also `outfolder` above.

`save` Enable saving of messages in the dead-letter file on interrupt or delivery error. See the variable **DEAD** for the location of the dead-letter file. The default shall be `save`.

`screen=`*number*
: Set the number of lines in a screenful of headers for the `headers` and `z` commands. If `screen` is not specified, a value based on the terminal type identified by the **TERM** environment variable, the window size, and/or the baud rate shall be used.

`showto` When the sender of the message was the user who is invoking `mailx`, write the information from the `"To:"` line instead of the `"From:"` line in the header summary. The default shall be `noshowto`.

`sign=`*string*
: Set the variable inserted into the text of a message when the `~a` command escape is given. The default shall be `nosign`. The character sequences `\t` and `\n` shall be recognized in the variable as `<tab>` and `<newline>`, respectively. (See also `~i` in 4.40.7.3).

`Sign=`*string*
: Set the variable inserted into the text of a message when the `~A` command escape is given. The default shall be `noSign`. The character sequences `\t` and `\n` shall be recognized in the variable as `<tab>` and `<newline>`, respectively. (See also `~i` in 4.40.7.3).

`toplines=`*number*
: Set the number of lines of the message to write with the `top` command. The default shall be 5.

4.40.7.2 `mailx` Commands

The following `mailx` commands shall be implemented. In the following list,
''header'' refers to lines from the message header, as shown in 4.40.6.3. ''Header- 7
line'' refers to lines within the header that begin with one or more non-white- 7
space characters, immediately followed by a colon and white space and continuing 7
until the next line beginning with a non-white-space character or an empty line. 7
''Header-field'' refers to the portion of a header line prior to the first colon in that
line.

For each of the commands listed below, the command can be entered as the abbreviation (those characters in the Synopsis command word preceding the [), the full command (all characters shown for the command word, omitting the [and]), or any truncation of the full command down to the abbreviation. For example, the `exit` command (shown as `ex[it]` in the Synopsis) can be entered as `ex`, `exi`, or `exit`.

The arguments to commands can be quoted, using the following methods:

— An argument can be enclosed between paired double-quotes (`" "`) or single-quotes (`' '`); any white space, shell word expansion, or backslash characters within the quotes shall be treated literally as part of the argument. A double-quote shall be treated literally within single-quotes and vice versa. These special properties of the quote marks shall occur only when they are paired at the beginning and end of the argument.

— A backslash outside of the enclosing quotes shall be discarded and the following character treated literally as part of the argument.

— An unquoted backslash at the end of a command line shall be discarded and the next line shall continue the command.

File names, where expected, shall be subjected to the process of shell word expan-
sions (see 3.6); if more than a single pathname results and the command is
expecting one file, the effects are unspecified. If the file name begins with an
unquoted plus-sign, it shall not be expanded, but treated as the named file (less 7
the leading plus) in the `folder` directory. (See the `folder` variable.) 7

4.40.7.2.1 Declare aliases

Synopsis : `a[lias]` [*alias* [*address* ...]] 7
Synopsis : `g[roup]` [*alias* [*address* ...]] 7

Add the given addresses to the alias specified by *alias* . The names shall be substi- 7
tuted when *alias* is used as a recipient address specified by the user in an outgo-
ing message (i.e., other recipients addressed indirectly through the `reply` com-
mand shall not be substituted in this manner). Mail address alias substitution
shall apply only when the alias string is used as a full address; e.g., when `hlj` is
an alias, `hlj@posix.com` does not trigger the alias substitution. If no arguments
are given, write a listing of the current aliases to standard output. If only an
alias argument is given, write a listing of the specified alias to standard output.
These listings need not reflect the same order of addresses that were entered.

4.40.7.2.2 Declare alternates

Synopsis : `alt[ernates]` *name* ...

Declare a list of alternate names for the user's login. When responding to a message, these names shall be removed from the list of recipients for the response. The comparison of names shall be in a case-insensitive manner. With no arguments, `alternates` shall write the current list of alternate names.

4.40.7.2.3 Change current directory

Synopsis : `cd [`*directory* `]`
Synopsis : `ch[dir] [`*directory* `]`

Change directory. If *directory* is not specified, the contents of **HOME** shall be used.

4.40.7.2.4 Copy messages

Synopsis : `c[opy] [`*file* `]`
Synopsis : `c[opy] [`*msglist* `]` *file*

Copy messages to the file named by the pathname *file* without marking the mes-
sages as saved. Otherwise, it shall be equivalent to the `save` command. 7

4.40.7.2.5 Delete messages

Synopsis : `d[elete] [`*msglist* `]`

Mark messages for deletion from the mailbox. The deletions shall not occur until
`mailx` quits (see 4.40.7.2.24) or changes mailboxes (see 4.40.7.2.10). If `auto-`
`print` is set, the next message after the last one deleted shall be written (see 8
4.40.7.1); if there is no subsequent message, the previous message, if it exists, 8
shall be written. In the case of no subsequent or previous message, or when 7
`noautoprint` is set, the `mailx` prompt shall be written. 7

4.40.7.2.6 Discard header fields

Synopsis : `di[scard] [`*header-field* `...]`
Synopsis : `ig[nore] [`*header-field* `...]` 7

Suppress the specified header fields when writing messages. Specified *header-* 7
fields shall be added to the list of suppressed header fields. Examples of header 7
fields to ignore are `status` and `cc`. The fields shall be included when the mes-
sage is saved. The `Print` and `Type` commands shall override this command. The
comparison of header fields shall be in a case-insensitive manner. If no argu- 7
ments are specified, write a list of the currently suppressed header fields to stan- 7
dard output; the listing need not reflect the same order of header fields that were 7
entered. 7

If both `retain` (see 4.40.7.2.27) and `discard` commands are given, `discard` 7
commands shall be ignored. 7

4.40.7.2.7 Delete messages and display

Synopsis : `dp` [*msglist*]
Synopsis : `dt` [*msglist*]

Delete the specified messages from the mailbox and write the next message after
the last one deleted. If there is no subsequent message, the `mailx` prompt shall 7
be written. 7

4.40.7.2.8 Edit messages

Synopsis : `e[dit]` [*msglist*]

Edit the given messages. The messages shall be placed in a temporary file and
the utility named by the **EDITOR** variable (see 4.40.5.3) shall be invoked to edit 7
the file. The default editor is unspecified. 7

4.40.7.2.9 Exit

Synopsis : `ex[it]`
Synopsis : `x[it]`

Exit from `mailx` without changing the mailbox. No messages shall be saved in the *mbox* (see also `quit`).

4.40.7.2.10 Change folder

Synopsis : `fi[le]` [*file*]
Synopsis : `fold[er]` [*file*]

Quit (see 4.40.7.2.24) from the current file of messages and read in the file named by the pathname *file* . If no argument is given, the name and status of the current mailbox shall be written.

Several unquoted special characters shall be recognized when used as *file* names, with the following substitutions:

%	The system mailbox for the invoking user. 7
%*user*	The system mailbox for *user* .
#	The previous file.
&	The current *mbox* .
+*file*	The named file in the `folder` directory. (See the `folder` variable.)

The default file shall be the current mailbox.

4.40.7.2.11 Display list of folders

Synopsis : `folders`

Write the names of the files in the directory set by the `folder` variable (see 4.40.5.3).

4.40.7.2.12 Display header summary for specified messages

Synopsis : `f[rom]` [*msglist*]

Write the header summary for the specified messages.

4.40.7.2.13 Display header summary

Synopsis : `h[eaders]` [*message*]

Write the page of headers that includes the message specified. The `screen` variable sets the number of headers per page. See also the `z` command.

4.40.7.2.14 Help

Synopsis : `hel[p]`
Synopsis : `?` 7

Write a summary of commands.

4.40.7.2.15 Hold messages

Synopsis : `ho[ld]` [*msglist*]
Synopsis : `pre[serve]` [*msglist*]

Mark the messages in *msglist* to be retained in the mailbox when `mailx` terminates. This shall override any commands that might previously have marked the messages to be deleted. During the current invocation of `mailx`, only the `delete`, `dp`, or `dt` commands shall remove the *preserve* marking of a message.

4.40.7.2.16 Execute commands conditionally

Synopsis : `i[f] s|r`
mail-command s
`el[se]`
mail-command s
`en[dif]`

Execute commands conditionally, where `if s` shall execute the following *mail-command* s, up to an `else` or `endif`, if the program is in Send Mode, and `if r` shall cause the *mail-command* s to be executed only in Receive Mode.

4.40.7.2.17 List available commands

Synopsis : `l[ist]`

Write a list of all commands available. No explanation shall be given.

4.40.7.2.18 Mail a message

Synopsis : `m[ail]` *address* . . .

Mail a message to the specified addresses or aliases.

4.40.7.2.19 Direct messages to mbox

Synopsis : `mb[ox]` [*msglist*]

Arrange for the given messages to end up in the *mbox* save file when `mailx` terminates normally. (See **MBOX** in 4.40.5.3 for a description of this file.) See also the `exit` and `quit` commands.

4.40.7.2.20 Process next specified message

Synopsis : `n[ext]` [*message*]
8

Go to the next message matching *message* . 8

4.40.7.2.21 Pipe message

Synopsis : `pi[pe]` [[*msglist*] *command*] 7
Synopsis : | [[*msglist*] *command*] 7

Pipe the messages through the given *command* by invoking the command inter- 7
preter specified by **SHELL** (see 4.40.5.3) with two arguments: `-c` and *command* . 7
(See also `sh -c` in 4.56.) The command shall be given as a single argument. Quoting, described previously, can be used to accomplish this. If no arguments are given, the current message shall be piped through the command specified by the value of the `cmd` variable. If the `page` variable is set, a `<form-feed>` character shall be inserted after each message.

4.40.7.2.22 Display message with headers

Synopsis : `P[rint]` [*msglist*]
Synopsis : `T[ype]` [*msglist*]

Write the specified messages, including all header lines, to standard output. Override suppression of lines by the `discard`, `ignore`, and `retain` commands. If `crt` is set, the messages longer than the number of lines specified by the `crt` variable shall be paged through the command specified by the **PAGER** environment variable. (See 4.40.5.3).

4.40.7.2.23 Display message

Synopsis : `p[rint]` [*msglist*]
Synopsis : `t[ype]` [*msglist*]

Write the specified messages to standard output. If `crt` is set, the messages longer than the number of lines specified by the `crt` variable shall be paged through the command specified by the **PAGER** environment variable. (See 4.40.5.3).

4.40.7.2.24 Quit

Synopsis : `q[uit]`
Synopsis : *end-of-file*

Terminate `mailx`, storing messages that were read in *mbox* (if the current mailbox is the system mailbox and unless `hold` is set), deleting messages that have been explicitly saved (unless `keepsave` is set), discarding messages that have been deleted, and saving all remaining messages in the mailbox.

4.40.7.2.25 Reply to a message list

Synopsis : `R[eply]` [*msglist*]
Synopsis : `R[espond]` [*msglist*]

Mail a reply message to the sender of each message in the *msglist* . The subject 7 line shall be formed by concatenating `"Re: "` (unless it already begins with that string) and the subject from the first message. If `record` is set to a file name, the response shall be saved at the end of that file.

See also the `flipr` variable in 4.40.7.1. 7

4.40.7.2.26 Reply to a message

Synopsis : `r[eply]` [*message*]
Synopsis : `r[espond]` [*message*]

Mail a reply message to all recipients included in the header of the message. The subject line shall be formed by concatenating `"Re: "` (unless it already begins with that string) and the subject from the message. If `record` is set to a file 7 name, the response shall be saved at the end of that file.

See also the `flipr` variable in 4.40.7.1. 7

4.40.7.2.27 Retain header fields

Synopsis : `ret[ain]` [*header-field* . . .]

Retain the specified header fields when writing messages. This command shall 7 override all `discard` and `ignore` commands. The comparison of header fields 7 shall be in a case-insensitive manner.

4.40.7.2.28 Save messages

Synopsis : `s[ave]` [*file*]
Synopsis : `s[ave]` [*msglist*] *file*

Save the specified messages in the file named by the pathname *file*, or the *mbox* if the *file* argument is omitted. The file shall be created if it does not exist; otherwise, the messages shall be appended to the file. The message shall be deleted from the mailbox when `mailx` terminates unless `keepsave` is set.

4.40.7.2.29 Set variables

Synopsis: `se[t]` [*name* [=[*string*]] . . .] [*name* =*number* . . .] [`no`*name* . . .] 7

Define one or more variables called *name*. The variable can be given a null, string, 7
or numeric value. Quoting and backslash escapes can occur anywhere in *string*,
as described previously, as if the *string* portion of the argument were the entire
argument. The forms *name* and *name*= shall be equivalent to *name* =`""` for vari- 7
ables that take string values. The `set` command without arguments shall write a 7
list of all defined variables and their values. The `no`*name* form shall be
equivalent to '`unset` *name*'.

4.40.7.2.30 Invoke a shell

Synopsis: `sh[ell]`

Invoke an interactive command interpreter (see also **SHELL** in 4.40.5.3).

4.40.7.2.31 Display message size

Synopsis: `si[ze]` [*msglist*]

Write the size in bytes of each of the specified messages. 7

4.40.7.2.32 Read `mailx` commands from a file

Synopsis: `so[urce]` *file*

Read and execute commands from the file named by the pathname *file* and return 7
to command mode.

4.40.7.2.33 Display beginning of messages

Synopsis: `to[p]` [*msglist*]

Write the top few lines of each of the specified messages. If the `toplines` vari- 7
able is set, it is taken as the number of lines to write. The default shall be 5.

4.40.7.2.34 Touch messages

Synopsis: `tou[ch]` [*msglist*]

Touch the specified messages. If any message in *msglist* is not specifically deleted 7
nor saved in a file, it shall be placed in the *mbox* upon normal termination. See 7
`exit` and `quit`.

4.40.7.2.35 Delete aliases

Synopsis: `una[lias]` [*alias*] . . .

Delete the specified alias names. If a specified alias does not exist, the results are unspecified.

4.40.7.2.36 Undelete messages

Synopsis : u[ndelete] [*msglist*]

Remove the deleted markings from the specified messages. If autoprint is set,
the last message of those restored shall be written. If *msglist* is not specified, it 7
shall default to the first deleted message following the current message that has 7
not been undeleted if there is one, or the last deleted message preceding the 7
current message that has not been undeleted otherwise. 7

4.40.7.2.37 Unset variables

Synopsis : uns[et] *name* ...

Cause the specified variables to be erased.

4.40.7.2.38 Edit message with full-screen editor

Synopsis : v[isual] [*msglist*]

Edit the given messages with a screen editor. The messages shall be placed in a
temporary file, and the utility named by the **VISUAL** variable (see 4.40.5.3) shall 7
be invoked to edit the file. The default editor shall be vi. 7

4.40.7.2.39 Write messages to a file

Synopsis : w[rite] [*msglist*] *file*

Write the given messages to the file specified by the pathname *file*, minus the 7
message header. Otherwise, it shall be equivalent to the save command.

4.40.7.2.40 Scroll header display

Synopsis : z[+|–]

Scroll the header display forward (if + is specified or if no option is specified) or 7
backward (if - is specified) one screenful. The number of headers written shall be 7
set by the screen variable. 7

4.40.7.2.41 Invoke shell command

Synopsis : !*command*

Invoke the command interpreter specified by **SHELL** (see 4.40.5.3) with two argu- 7
ments: –c and *command* . (See also sh –c in 4.56.) If the bang variable is set, 7
each unescaped occurrence of ! in command shall be replaced with the command 7
executed by the previous ! command or ~! command escape. 7

4.40.7.2.42 Null command

Synopsis : # *comment*

This null command (comment) shall be ignored by mailx.

4.40.7.2.43 Display current message number

Synopsis : `=`

Write the current message number.

7

4.40.7.3 `mailx` Command Escapes

The following commands can be entered only from input mode, by beginning a line with the escape character [by default, tilde (~)]. See the `escape` variable description for changing this special character. The format for the commands is:

<ESC><command-char><separator> [*<argument(s)>*]

where the *<separator>* can be zero or more `<blank>`s.

In the following descriptions, the argument *command* (but not *mailx-command)* shall be a shell command string. Any string acceptable to the command inter- 7
preter specified by the **SHELL** variable (see 4.40.5.3) when it is invoked as `–c` 7
command_string (see 4.56 for a description of `sh –c`) shall be valid. The command 7
can be presented as multiple arguments (i.e., quoting is not required).

Command escapes that are listed with *msglist* or *mailx-command* arguments are invalid in Send Mode and produce unspecified results.

~! *command* — Invoke the command interpreter specified by **SHELL** (see 7
4.40.5.3) with two arguments: `–c` and *command* ; and then 7
return to input mode. If the `bang` variable is set, each unes- 7
caped occurrence of ! in *command* shall be replaced with the 7
command executed by the previous ! command or ~! com- 7
mand escape. 7

`~.` — Simulate end-of-file (terminate message input).

`~:` *mailx-command*
`~_` *mailx-command*
— Perform the command-level request.

`~?` — Write a summary of command escapes.

`~A` — This shall be equivalent to `~i Sign`.

`~a` — This shall be equivalent to `~i sign`.

`~b` *name* ... — Add the *name* s to the blind carbon copy (Bcc) list.

`~c` *name* ... — Add the *name* s to the carbon copy (Cc) list.

`~d` — Read in the dead-letter file. See **DEAD** (4.40.5.3) for a description of this file.

`~e` — Invoke the editor, as specified by the **EDITOR** environment variable (see 4.40.5.3), on the partial message.

`~f` [*msglist*] Forward the specified messages. The specified messages shall be inserted into the current message without alteration. This command escape shall also insert message headers into the message with field selection affected by the `discard`, `ignore`, and `retain` commands.

`~F` [*msglist*] This shall be the equivalent of the `~f` command escape, except that all headers shall be included in the message, regardless of previous `discard`, `ignore`, and `retain` commands.

`~h` If standard input is a terminal, prompt for a Subject line and the To, Cc, and Bcc lists. Other implementation-defined headers may also be presented for editing. If the field is written with an initial value, it can be edited as if it had just been typed.

`~i` *string* Insert the value of the named variable, followed by a `<newline>`, into the text of the message. If the string is unset or
null, the message shall not be changed. 7

`~m` [*msglist*] Insert the specified messages into the message, prefixing nonempty lines with the string in the `indentprefix` variable. This command escape shall also insert message headers into the message, with field selection affected by the `discard`, `ignore`, and `retain` commands.

`~M` [*msglist*] This shall be the equivalent of the `~m` command escape, except that all headers shall be included in the message, regardless of previous `discard`, `ignore`, and `retain` commands.

`~p` Write the message being entered. If the message is longer 7
than `crt` lines (see 4.40.7.1), the output shall be paginated 7
as described for the **PAGER** variable (see 4.40.5.3). 7

`~q` Quit (see 4.40.7.2.24) from input mode by simulating an interrupt; see 4.40.5.4. If the body of the message is not empty, the partial message shall be saved in the dead-letter file. See **DEAD** (4.40.5.3) for a description of this file.

`~r` *file*
`~<` *file*
`~<` `!`*command* Read in the file specified by the pathname *file*. If the argument begins with an exclamation-mark (`!`), the rest of the
string shall be taken as an arbitrary system command; the 7
command interpreter specified by **SHELL** (see 4.40.5.3) shall 7
be invoked with two arguments: `−c` and *command* . (See also 7
`sh −c` in 4.56.) The standard output of *command* shall be 7
inserted into the message. 7

`~s` *string* Set the subject line to *string* .

`~t` *name* ... Add the given *name* s to the To list.

`~v` Invoke the full-screen editor, as specified by the **VISUAL** environment variable (see 4.40.5.3), on the partial message.

`~w` *file* Write the partial message, without the header, onto the file named by the pathname *file*. The file shall be created or the message shall be appended to it if the file exists.

`~x` Exit as with `~q`, except the message shall not be saved in the dead-letter file.

`~|` *command* Pipe the body of the message through the given *command* by invoking the command interpreter specified by **SHELL** (see 4.40.5.3) with two arguments: `-c` and *command*. If the *command* returns a successful exit status, the standard output of the command shall replace the message. Otherwise the message shall remain unchanged. If the *command* fails, an error message giving the exit status shall be written.

4.40.8 Exit Status

When the `-e` option is specified, the `mailx` utility shall exit with one of the following values:

- 0 Mail was found.
- >0 Mail was not found or an error occurred.

Otherwise, `mailx` utility shall exit with one of the following values:

- 0 Successful completion.
- >0 An error occurred.

4.40.9 Consequences of Errors

When in input mode (Receive Mode) or Send Mode:

- — If an error is encountered processing a command escape (see 4.40.7.3), a diagnostic message shall be written to standard error, and the message being composed may be modified, but this condition shall not prevent the message from being sent.
- — Other errors shall prevent the sending of the message.

When in command mode:

- — Default.

4.40.10 Rationale. *(This subclause is not a part of P1003.2a)*

Usage, Examples

The format of the system mailbox is intentionally unspecified. Not all systems will implement system mailboxes as flat files, particularly with the advent of multimedia mail messages. Some system mailboxes may be multiple files, others records in a database. The internal format of the messages themselves are specified with the historical format from Version 7, but only after they have been saved in some file other than the system mailbox. This was done so that many historical applications expecting text-file mailboxes will not be broken.

Some new formats for messages can be expected in the future, probably including binary data, bit maps, and various multimedia objects. As described here, `mailx` is not prohibited from handling such messages, but it must store them as text files in secondary mailboxes (unless some extension, such as a variable or command-line option, is used to change the stored format). Its method of doing so is implementation defined and might include translating the data into text-file-compatible or readable form or omitting certain portions of the message from the stored output.

Some historical implementations support the following internal variable:

`debug` Enable verbose diagnostics for debugging. Messages shall not be delivered. The default shall be `nodebug`.

The `edit` and `visual` commands merely edit the specified messages in a temporary file. They do not modify the contents of those messages in the mailbox; such a capability could be added as an extension, such as by using different command names.

The `discard`/`ignore` and `retain` commands are not inverses of each other. The `retain` command discards all header-fields except those explicitly retained. The `discard` command keeps all header-fields except those explicitly discarded. After a `retain` command is specified, `discard` and `ignore` commands are ignored.

The restriction on a subject line being {LINE_MAX} – 10 bytes is based on the historical format that consumes 10 bytes for `"Subject: "` and the trailing `<newline>`. Many historical mailers that a message may encounter on other systems will not be able to handle lines that long, however.

History of Decisions Made

Any references to networking have been removed as being outside the scope of the document.

The developers of the standard felt strongly that a method for applications to send messages to specific users was necessary. The obvious example is a batch utility, running noninteractively, that wishes to communicate errors or results to a user. However, the actual format, delivery mechanism, and method of reading the message are clearly beyond the scope of this standard.

The intent of this command is to provide a simple, portable interface for sending messages noninteractively. It merely defines a ''front-end'' to the historical mail system. It is suggested that implementations explicitly denote the sender and recipient in the body of the delivered message. Further specification of formats for either the message envelope or the message itself were deliberately not made, as the industry is in the midst of changing from the current standards to a more internationalized standard and it is probably incorrect, at this time, to require either one.

Implementations are encouraged to conform to the various delivery mechanisms desribed in the CCITT X.400 standards or to the equivalent Internet Standards described in Internet ''Request for Comment'' (RFC) documents RFC 819, RFC 822, RFC 920, RFC 921, and RFC 1123. Copies of these Internet RFCs can be obtained from the DDN Network Information Center, SRI International, Menlo Park, CA, USA 94025. *Editor's Note: This paragraph will be reworked editorially to use a bibliographic entry after the two standards are merged. Doing so now is rather a headache.*

The standard does not place any restrictions on the length of messages handled by `mailx`, and for delivery of local messages the only limitations should be the normal problems of available disk space for the target mail file. When sending messages to external machines, applications are advised to limit messages to less than 50 kilobytes because many mail gateways impose message-length restrictions. (Note that this is usually an administrative issue based on the amount of mail traffic and disk space available on the gateways. Therefore, there is no way for this standard to require implementations to guarantee delivery of long messages to remote systems.)

Like the utilities `logger` and `lp`, `mailx` is admittedly difficult to test. This was not deemed sufficient justification to exclude these utilities from the standard. It is also arguable that they are, in fact, testable, but that the tests themselves are not portable.

When `mailx` is being used by an application that wishes to receive the results as if none of the User Portability Utilities Option features were supported, the **DEAD** environment variable must be set to `/dev/null`. Otherwise, it may be subject to the file creations described in 4.40.5.4. Similarly, if the **MAILRC** environment variable is not set to `/dev/null`, historical versions of `mailx` and `Mail` read initialization commands from a file before processing begins. Since the initialization that a user specifies could alter the contents of messages an application is trying to send, such applications must set **MAILRC** to `/dev/null`.

The description of **LC_TIME** uses ''may affect'' because many historical implementations do not or cannot manipulate the date and time strings in the incoming mail headers. Some headers found in incoming mail do not have enough information to determine the time zone in which the mail originated, and, therefore, `mailx` cannot convert the date and time strings into the internal form that can then be parsed by routines like *strftime*() that can take **LC_TIME** settings into account. Changing all these times to a user-specified format is allowed, but not required.

The paginator selected when **PAGER** is null or unset is partially unspecified to allow System V historical practice of using `pg` as the default. Bypassing the pagination function, such as by declaring that `cat` is the paginator, would not meet with the intended meaning of this description. However, any ''portable user'' would have to set **PAGER** explicitly to get his or her preferred paginator on all systems. The paginator choice was made partially unspecified, unlike the **VISUAL** editor choice (mandated to be `vi`) because most historical pagers follow a common theme of user input, whereas editors differ dramatically.

Options to specify addresses as ''cc''(carbon-copy) or ''bcc''(blind-carbon-copy) were considered to be format details and were omitted.

A zero exit status implies that all messages were *sent* , but it gives no assurances that any of them were actually *delivered* . The reliability of the delivery mechanism is unspecified and is an appropriate marketing distinction between systems.

In order to conform to the utility syntax guidelines, a solution was required to the optional *file* option-argument to `−f`. By making *file* an operand, the guidelines are satisfied and users remain portable. However, it does force implementations to support usage such as:

```
mailx -fin mymail.box
```

The `−F` was incorrectly stated in previous drafts to apply to Receive Mode, but it has only been in `mailx` Send Mode historically. It, along with the `Copy`, `Save`, `followup`, and `Followup` commands, has been omitted. Historical systems used these to save messages in a file whose name is ''derived from the author (or recipient) of the message.'' Because of some unusual network mail addresses currently in use, and because POSIX does not specify anything about the content of the mail headers, it is not possible for this standard to identify the algorithm used for this derivation; since files would therefore be created in an unpredictable manner, this is unacceptable for portable usage.

The `no`*name* method of unsetting variables is not present in all historical systems, but it is in System V and provides a logical set of commands corresponding to the format of the display of options from the `mailx set` command without arguments.

The `ask` and `asksub` variables are the names selected by BSD and System V, respectively, for the same feature. They are synonyms here.

The `mailx echo` command was not documented in the BSD version and has been omitted here because it is not obviously useful for interactive users.

The default prompt on the System V `mailx` is a question mark, on BSD `Mail` an ampersand. Since POSIX.2 chose the `mailx` name, it kept the System V default, assuming that BSD users would not have difficulty with this minor incompatibility (that they can override anyway).

The meanings of `r` and `R` are reversed between System V `mailx` and Sun OS `Mail`. Once again, since POSIX.2 chose the `mailx` name, it kept the System V default, but allows the Sun OS user to achieve the desired results using `flipr`, an internal variable in System V `mailx`, although it has not been documented in the *SVID*.

The `indentprefix` variable, the `retain` and `unalias` commands, and the `~F` and `~M` command escapes were adopted from 4.3BSD `Mail`.

The `version` command was not included because no sufficiently general specification of the version information could be devised that would still be useful to a portable user. This command name should be used by suppliers who wish to provide version information about the `mailx` command.

The ''implementation-specific (unspecified) system startup file'' has been traditionally named `/etc/mailx.rc`, but this specific name and location are not required.

4.52 `read` — Read a line from standard input

⇒ **4.52.2 `read` Description.** *In subclause 4.52, add the following to the end of the second paragraph in the Description subclause:*

If standard input is a terminal device and the invoking shell is interactive, `read` shall prompt for a continuation line when:

— The shell reads an input line ending with a backslash, unless the `−r` option is specified.

— A here-document is not terminated after a `<newline>` is entered.

⇒ **4.52.5.3 `read` Environment Variables.** *In subclause 4.52, add the following environment variable in the correct sorted order to the Environment Variables subclause.*

PS2 This variable shall contain the prompt string that an interactive shell shall write to standard error when a line ending with a backslash is read and the `−r` option was not specified, or if a here-document is not terminated after a `<newline>` is entered.

⇒ **4.51.6.2 `read` Standard Error.** *In subclause 4.52, replace the Standard Error subclause with the following.*

Used for diagnostic messages and prompts for continued input.

Rationale. *(This subclause is not a part of P1003.2a)*

None.

4.56 sh — Shell, the standard command language interpreter

⇒ **4.56.1 sh Synopsis.** *Add the* –b*,* –m*, and* –o *option options to each of the
three Synopsis lines with appropriate editorial wording indicating that these* 7
options need to be supported only when the UPUO is supported: 7

⇒ **4.56.3 sh Options.** *Change the first sentence of the second paragraph, beginning* 'The –a, –C, . . . '' *to:*

The –a, –b, –C, –e, –f, –m, –n, –o *option* , –u, –v, and –x options are described as part of the set utility in 3.14.11.

⇒ **4.56.3 sh Options.** *Remove the following sentence and NOTE:* 8

The behavior of an interactive shell is not fully specified by this standard. 8

NOTE: The preceding sentence is expected to change following the eventual approval of the UPE 8
supplement. 8

⇒ **4.56.5.3 sh Environment Variables.** *Modify the contents of subclause 4.56.5.3, Environment Variables, to add the following environment variables in the correct sorted order.*

FCEDIT 8
HISTFILE 8
HISTSIZE See the description of the fc environment variables in 8
5.12.5.3. This standard specifies the effects of these 8
variables only for systems supporting the User Porta- 8
bility Utilities Option. 8

MAIL This variable shall be interpreted as a pathname of the user's mailbox file for purposes of incoming mail notification. If this variable is set, the shell shall inform the user if the file named by the variable is created or if its modification time has changed. Informing the user shall be accomplished by writing a string of unspecified format to the standard error prior to the writing of the next primary prompt string after the completion of an interval defined by the **MAIL-CHECK** variable. The user shall be informed only if
MAIL is set and **MAILPATH** is not set. This standard 7

specifies the effects of this variable only for systems 7
supporting the User Portability Utilities Option. 7

MAILCHECK This variable shall be interpreted as a decimal integer value that specifies how often (in seconds) the shell shall check for the arrival of mail in the files specified by the **MAILPATH** or **MAIL** variables. The default value shall be 600 seconds. If set to zero, the shell
shall check before issuing each primary prompt. This 7
standard specifies the effects of this variable only for 7
systems supporting the User Portability Utilities 7
Option. 7

MAILPATH This variable shall be interpreted as a list of pathnames and optional messages separated by colons. If this variable is set, the shell shall inform the user if any of the files named by the variable are created or if any of their modification times change. (See the preceding entry for **MAIL** for descriptions of mail arrival and user informing.) Each pathname can be
followed by `%` and a string that shall be subjected to 7
parameter expansion and written to standard error 7
when the modification time changes. If a `%` character in the pathname is preceded by a backslash, it shall be treated as a literal `%` in the pathname. The default
message is unspecified. This standard specifies the 7
effects of this variable only for systems supporting the 7
User Portability Utilities Option. 7

The **MAILPATH** environment variable shall take precedence over the **MAIL** variable.

⇒ **4.56.7 `sh` Extended Description.** *Replace the subclause 4.56.7, Extended Description, with the following.*

4.56.7 Extended Description

See the description of the shell command language in Section 3.

On systems supporting the User Portability Utilities Option, the following additional capabilities shall be supported.

4.56.7.1 Command History List 8

When the `sh` utility is being used interactively, it shall maintain a list of com- 8
mands previously entered from the terminal in the file named by the **HISTFILE** 8
environment variable. The type, size, and internal format of this file are 8
unspecified. Multiple `sh` processes can share access to the file for a user, if file 8

access permissions allow this; see the description of the **HISTFILE** environment 8
variable in 5.12.5.3. 8

4.56.7.2 Command Line Editing

When `sh` is being used interactively from a terminal, the current command and
the command history (see `fc` in 5.12) can be edited using *vi-mode* command-line 8
editing. This mode uses commands, described below, similar to a subset of those 8
described in the `vi` (see 5.35) utility. Implementations may offer other 8
command-line editing modes corresponding to other editing utilities. 8

The command `set -o vi` shall enable vi-mode editing and place `sh` into `vi` 8
insert mode (see 4.56.7.3). This command shall also disable any other editing 8
mode that the implementation may provide. The command `set +o vi` shall dis- 8
able vi-mode editing. 8

Certain block-mode terminals may be unable to support shell command line editing. If a terminal is unable to provide either edit mode, it need not be possible to
`set -o vi` when using the shell on this terminal. 7

In the following subclauses, the characters *erase*, *interrupt*, *kill*, and *end-of-file* are those set by the `stty` utility; see 4.59.

4.56.7.3 Vi-Mode Command Line Editing

With vi-mode enabled, `sh` can be switched between insert mode and command mode.

When in insert mode, an entered character shall be inserted into the command
line, except as noted in 4.56.7.4. Upon entering `sh` and after termination of the 7
previous command, `sh` shall be in insert mode. 7

Typing `<ESC>` shall switch `sh` into command mode (see 4.56.7.5). In command
mode, an entered character shall either invoke a defined operation, be used as
part of a multicharacter operation, or be treated as an error. A character that is
not recognized as part of an editing command shall terminate any specific editing
command and shall alert the terminal. Typing the *interrupt* character in com- 8
mand mode shall cause `sh` to terminate command-line editing on the current com- 8
mand line, re-issue the prompt on the next line of the terminal, and reset the com- 8
mand history (see 5.12) so that the most recently executed command is the ‘‘previ- 8
ous’’ command (i.e., the command that was being edited when it was interrupted 8
is not re-entered into the history). 8

In the following subclauses, the phrase ‘‘move the cursor to the beginning of the
word’’ shall mean ‘‘move the cursor to the first character of the current word’’ and
the phrase ‘‘move the cursor to the end of the word’’ shall mean ‘‘move the cursor
to the last character of the current word.’’ The phrase ‘‘beginning of the command 8
line’’ indicates the point between the end of the prompt string issued by the shell
(or the beginning of the terminal line, if there is no prompt string) and the first
character of the command text.