IEEE P1003.2a Draft 8 – December 1991

PS 1 Used for interactive prompts. Historically, the ‘‘super-user’’ has 7
had a prompt of `"# "`. Since privileges are not required to be 7
monolithic, it is difficult to define which privileges should cause 7
the alternate prompt. However, a sufficiently powerful user 7
should be reminded of that power by having an alternate prompt. 7

PS 2 Used for interactive prompts.

PS 3 Used by the KornShell for the `select` command. Since the POSIX.2 shell does not support `select`, **PS 3** was omitted.

PS 4 Used for shell debugging. For example, the following script:

```
PS4='[${LINENO}]+ '
set -x
echo Hello
```

writes the following to standard error:

```
[3]+ echo Hello
```

RANDOM This pseudo-random number generator was not seen as being very useful to interactive users.

SECONDS Although this variable is sometimes used with **PS 1** to allow the display of the current time in the user's prompt, it is not one that would be manipulated frequently enough by an interactive user to include in POSIX.

3.12 Shell Execution Environment

⇒ **3.12 Shell Execution Environment.** *Add another item to the list of the contents of the shell execution environment:*

— Shell aliases (see 3.3.1).

3.14.11 `set` — Set/unset options and positional parameters

⇒ **3.14.11 `set`.** *Merge the following options into the the first two synopsis lines of* 7
subclause 3.14.11 with appropriate editorial wording indicating that these 7
options need to be supported only when the UPUO is supported: 7

`set` [`-bm`] [`-o` *option*] [*argument* ...] 7
`set` [`+bm`] [`+o` *option*] [*argument* ...] 7

⇒ **3.14.11 `set`.** *Modify the contents of subclause 3.14.11 to add the following options in the correct sorted order.*

`–b` Cause the shell to notify the user asynchronously of background job completions. The following message shall be written to standard error:

`"[%d]%c %s%s\n"`, *<job-number>* , *<current>* , *<status>*, *<job-name>*

where the fields shall be as follows:

<current>
The character `+` identifies the job that would be used as a default for the `fg` or `bg` utilities; this job can also be specified using the *job_id* `%+` or `%%`. The character `–` identifies the job that would become the default if the current default job were to exit; this job can also be specified using the *job_id* `%–`. For other jobs, this field is a `<space>`. At most one job can be identified with `+` and at most one job can be identified with `–`. If there is any suspended job, then the current job shall be a suspended job. If there are at least two suspended jobs, then the previous job shall also be a suspended job.

<job-number>
A number that can be used to identify the process group to the `wait`, `fg`, `bg`, and `kill` utilities. Using these utilities, the job can be identified by prefixing the job number with `%`.

<status>
Unspecified.

<job-name>
Unspecified.

When the shell notifies the user a job has been completed, it may remove the job's process ID from the list of those ‘‘known in the current shell execution environment’’; see 3.9.3.1. Asynchronous notification shall not be enabled by default.

`−m` This option shall be supported if the system supports both the User Portability Utilities Option and the POSIX.1 {8} Job Control Option. All jobs shall be run in their own process groups. Immediately before the shell issues a prompt after completion of the background job, a message reporting the exit status of the background job shall be written to standard error. If a foreground job stops, the shell shall write a message to standard output to that effect, formatted as described by the `jobs` utility (see 5.15.6.1). In addition, if a job changes status other than exiting (for example, if it stops for input or output, or is stopped by a SIGSTOP signal), the shell shall write a similar message immediately prior to writing the next prompt. On systems supporting the POSIX.1 {8} Job Control Option, this option shall be enabled by default for interactive shells.

`−o` *option*

Set various options, many of which shall be equivalent to the single option letters. The following values of *option* shall be supported:

`allexport` Equivalent to `−a`.

8

`errexit` Equivalent to `−e`.

`ignoreeof` Prevent an interactive shell from exiting on end-of- 7
file. 7

`monitor` This option-argument shall be supported if the system supports both the User Portability Utilities Option and the POSIX.1 {8} Job Control Option. It shall be equivalent to the `−m` option.

`noclobber` Equivalent to `−C` (uppercase C).

`noglob` Equivalent to `−f`.

`noexec` Equivalent to `−n`.

`notify` Equivalent to `−b`.

`nounset` Equivalent to `−u`.

`verbose` Equivalent to `−v`.

`vi` Allow shell command-line editing using the built-in
`vi` editor. Enabling `vi` mode shall disable any 8
other command-line editing mode provided as an 8
implementation extension. 8

It need not be possible to set `vi` mode on for certain block-mode terminals.

`xtrace` Equivalent to `−x`.

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

The `ignoreeof` setting prevents accidental logouts when the end-of-file character (typically `control-D`) is entered. A user must explicitly `exit` to leave the interactive shell.

The `–o` option was originally in the base POSIX.2 document, but was removed because it is not commonly used by scripts and all of the options required for application portability had (or were given) single-letter option names. However, users have different needs and the `–o` option was adopted from the KornShell to address these. In addition to its generally friendly interface, `–o` is needed to provide the `vi` command-line editing mode, for which historical practice yields no single-letter option name. (Although it might have possible to invent such a letter, it was recognized that other editing modes would be developed and `–o` provides ample name space for describing such extensions.)

It is not possible to use the `$-` special parameter to determine the current setting of either of these two modes. In the KornShell, the *option* option-argument can be omitted, producing a report of current option settings. Since it breaks the utility syntax guidelines, and since the output format was unspecified (it changed between KornShell versions), this usage was omitted.

The `set –m` option was added to apply only to the UPE because it applies primarily to interactive use, not shell script applications.

The new `set –o` option-arguments apply only to the UPE because they are needed to enable the line editing modes to be used in the additions to `sh` for the UPE.

The ability to do asynchronous notification became available in the 1988 version of the KornShell. To have it occur, the user had to issue the command

```
trap "jobs -n" CLD
```

The C-shell provides two different levels of an asynchronous notification capability. The environment variable **notify** is analogous to what is done in `set -b` or `set -o notify`. When set, it notifies the user immediately of background job completions. When unset, this capability is turned off.

The other notification ability comes through the built-in utility `notify`. The syntax is

```
notify [%job ...]
```

By issuing `notify` with no operands, it causes the C-shell to notify the user asynchronously when the state of the current job changes. If given operands, `notify` will asynchronously inform the user of changes in the states of the specified jobs.

To add asynchronous notification to the POSIX shell, neither the KornShell's extensions to `trap`, nor the C-shell's **notify** environment variable seemed appropriate (the **notify** variable is not a proper POSIX environment variable name).

The `set -b` option was selected as a compromise.

The `notify` built-in was considered to have more functionality than was required for simple asynchronous notification.

Section 4: Revisions to Execution Environment Utilities

4.12 `command` — Execute a simple command

⇒ **4.12.1 `command` Synopsis.** *Add the following lines to the* `command` *utility Synopsis:*

On systems supporting the User Portability Utilities Option, the following 7
forms shall also be supported: 7

`command [ −v | −V ]` *command_name*

⇒ **4.12.2 `command` Description.** *Add the following paragraph to the end of the* `command` *utility Description:*

On systems supporting the User Portability Utilities Option, the `command` utility shall also provide information concerning how a command name will be interpreted by the shell; see `−v` and `−V`.

⇒ **4.12.3 `command` Options.** *Add the following options in the correct sorted order:*

`−v` Write a string to standard output that indicates the pathname
or command that will be used by the shell, in the current shell 7
execution environment (see 3.12), to invoke *command_name* . 7

— Utilities, regular built-in utilities, *command_names* including a slash character, and any implementation-provided functions that are found using the **PATH** variable (as described in 3.9.1.1), shall be written as absolute pathnames.

— Shell functions, special built-in utilities, regular built-in 7
utilities not associated with a **PATH** search, and shell 7
reserved words shall be written as just their names. 7

— An alias shall be written as a command line that represents its alias definition.

— Otherwise, no output shall be written and the exit status shall reflect that the name was not found.

`−V` Write a string to standard output that indicates how the name given in the *command_name* operand will be interpreted by
the shell, in the current shell execution environment (see 7
3.12). Although the format of this string is unspecified, it 7
shall indicate in which of the following categories *command_name* falls and include the information stated:

— Utilities, regular built-in utilities, and any implementation-provided functions that are found using the **PATH** variable (as described in 3.9.1.1), shall be identified as such and include the absolute pathname in the string.

— Other shell functions shall be identified as functions.

— Aliases shall be identified as aliases and their definitions shall be included in the string.

— Special built-in utilities shall be identified as special built-in utilities.

— Regular built-in utilities not associated with a **PATH** 7
search shall be identified as regular built-in utilities. (The 7
term ‘‘regular’’ need not be used.) 7

— Shell reserved words shall be identified as reserved words.

⇒ **4.12.6.1 `command` Standard Output.** *Change* ‘None.’’ *to:*

When the `−v` option is specified, standard output shall be formatted as:

`"%s\n"`, <*pathname or command*>

When the `−V` option is specified, standard output shall be formatted as:

`"%s\n"`, <*unspecified*>

⇒ **4.12.8 `command` Exit Status.** *Change the first sentence, beginning* ‘The command utility shall exit . . . ’’*to*

When the `−v` or `−V` options are specified, the `command` utility shall exit with 8
one of the following values:

0 Successful completion.

>0 The *command_name* could not be found or an error occurred.

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

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

The `command` –`v` and –`V` options were added to the UPE to satisfy requirements from users that are currently accomplished by three different historical utilities: `type` in the System V shell, `whence` in the KornShell, and `which` in the C-shell. Since there is no historical agreement on how and what to accomplish here, the POSIX `command` utility was enhanced and the historical utilities were left unmodified. The C-shell `which` merely conducts a path search. The KornShell `whence` is more elaborate—in addition to the categories required by POSIX, it also reports on tracked aliases, exported aliases, and undefined functions.

Since the –`v` and –`V` options of `command` produce output in relation to the current 7
shell execution environment, `command` is generally provided as a shell regular 7
built-in. If it is called in a subshell or separate utility execution environment, 7
such as one of the following: 7

```
(PATH=foo command -v)
nohup command -v
```
7 7

it will not necessarily produce correct results. For example, when called with 7
`nohup` or an *exec* function, in a separate utility execution environment, most 7
implementations will not be able to identify aliases, functions, or special built-ins. 7

Two types of regular built-ins could be encountered on a system and these are 7
described separately by `command`. The description of command search in 3.9.1.1 7
allows for a standard utility to be implemented as a regular built-in as long as it 7
is found in the appropriate place in a **PATH** search. So, for example, 7
`command -v true` might yield `/bin/true` or some similar pathname. Other 7
implementation-provided utilities that are not defined by POSIX.2 might exist only 7
as built-ins and have no pathname associated with them. These will produce out- 7
put identified as (regular) built-ins. Applications encountering these will not be 7
able to count on *exec*ing them, using them with `nohup`, overriding them with a 7
different **PATH**, etc. 7

The output format of –`V` was left mostly unspecified because human users are its only audience. Applications should not be written to care about this information; they can use the output of –`v` to differentiate between various types of commands, but the additional information that may be emitted by the more verbose –`V` is not needed and should not be arbitrarily constrained in its verbosity or localization for application parsing reasons.

4.32 `kill` — Terminate or signal processes

⇒ **4.32.4 `kill` Operands.** *In clause 4.32, modify the description of the pid operand in the Operands subclause:*

pid One of the following:

(1) *<current contents>*

(2) On systems supporting both the POSIX.1 {8} job control option and the User Portability Utilities Option, a job control job ID (see 2.2.2.206) that identifies a background process group to be signaled. The job control job ID notation is applicable only for invocations of `kill` in the current shell execution environment; see 3.12.

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

The job control job ID notation is not required to work as expected when `kill` is operating in its own utility execution environment. In either of the following examples:

```
nohup kill %1 &
system("kill %1");
```

the `kill` operates in a different environment and will not share the shell's understanding of job numbers.

4.40 `mailx` — Process messages

Editor's Note: This clause contains text that is directly extracted from POSIX.2 8
Draft 11.2 and, as such, is not eligible for balloting by the POSIX.2a group. This 8
text is marked with the '1'' or '2'' diff-marks. 8

⇒ **4.40 `mailx`.** *Replace the entire contents of clause 4.40 with:*

4.40.1 Synopsis

Send Mode:

`mailx [−s` *subject* `]` *address* ...

On systems supporting the User Portability Utilities Option, the following forms shall also be valid (Receive Mode):

`mailx −e`

`mailx [−HiNn][−u` *user* `]` 7

`mailx −f [−HiNn] [`*file* `]`

4.40.2 Description

The `mailx` utility provides a message sending and receiving facility.

When this standard is implemented without the User Portability Utilities Option,
an application using `mailx` shall have the ability to send messages directly from
standard input to one or more addresses in an unspecified manner. (Send Mode.) 1
Unless the first character of one or more lines is tilde (~), all characters in the 1
input message shall appear in the delivered message, but additional characters 1
may be inserted in the message before it is retrieved. 1

On systems supporting the User Portability Utilities Option, mail-receiving capabilities and other interactive features, Receive Mode, described below, shall also be enabled.

When reading mail, `mailx` provides commands to facilitate saving, deleting, and responding to messages. When sending mail, `mailx` allows editing, reviewing, and other modification of the message as it is entered.

Incoming mail shall be stored in one or more unspecified locations for each user, 7
collectively called the system *mailbox* for that user. When `mailx` is invoked in 7
Receive Mode, the system mailbox shall be the default place to find them. As
messages are read, they shall be marked to be moved to a secondary file for
storage, unless specific action is taken. This secondary file is called the *mbox* and 7
is normally located in the **HOME** directory of the user (see **MBOX**, in 4.40.5.3, for
a description of this file). Messages shall remain in this file until explicitly

removed. When the `-f` option is used to read mail messages from secondary files, messages shall be retained in those files unless specifically removed. All three of these locations—system mailbox, *mbox*, and secondary file—are referred to in this clause as simply ''mailboxes,'' unless more specific identification is required.

4.40.3 Options

The `mailx` utility shall conform to the utility argument syntax guidelines described in 2.10.2.

The following options shall be supported by the implementation. Only the `-s` *subject* option shall be required on all systems. The other options are required only on systems supporting the User Portability Utilities option.

`-e` Test for the presence of mail in the system mailbox. The `mailx` 7
utility shall write nothing and exit with a successful return code if there is mail to read.

`-f` Read messages from the file named by the *file* operand instead of the system mailbox. (See also `folder`). If no *file* operand is specified, read messages from the *mbox* instead of the system mailbox.

`-H` Write a header summary only.

`-i` Ignore interrupts. (See also `ignore`).

`-n` Do not initialize from the system default startup file. See 4.40.7.

`-N` Do not write an initial header summary.

`-s` *subject* Set the Subject header field to *subject*. All characters in the *sub-* 2
ject string shall appear in the delivered message. The results are 2
unspecified if *subject* is longer than {LINE_MAX} – 10 bytes or con- 2
tains a `<newline>`. 2

`-u` *user* Read the system mailbox of the login name *user*. This shall only
be successful if the invoking user has the appropriate privileges 7
to read the system mailbox of that user.

4.40.4 Operands

The following operands shall be supported by the implementation:

address Addressee of message. When `-n` is specified and no user startup 7
files are accessed (see 4.40.7), this shall be an address to pass to 7
the mail delivery system. Any system or user startup files may 7
enable aliases (see 4.40.7.2.1), that may modify the form of 7
address before it is passed to the mail delivery system. 7

file A pathname of a file to be read instead of the system mailbox when `-f` is specified. The meaning of the *file* option-argument shall be affected by the contents of the `folder` internal variable; see 4.40.7.1.

4.40.5 External Influences

4.40.5.1 Standard Input

When `mailx` is invoked in Send Mode (the first synopsis line), standard input shall be the message to be delivered to the specified addresses. If the User Portability Utilities Option is not supported, standard input lines beginning with a tilde (~) character produce unspecified results.

When `mailx` is invoked in Receive Mode, (the second to fourth synopsis lines), standard input can contain escaped commands (by default, lines beginning with tilde), as described in 4.40.7.

4.40.5.2 Input Files

When `mailx` is used as described by this standard, the *file* option-argument (see the `-f` option) and the *mbox* shall be text files containing mail messages, formatted as described in 4.40.6.3. The nature of the system mailbox is unspecified; it need not be a file.

4.40.5.3 Environment Variables

The following environment variables shall affect the execution of `mailx`:

DEAD This variable shall be interpreted as a pathname of the file in which to save partial messages in case of interrupts or delivery errors. The default shall be `dead.letter` in the directory named by the **HOME** variable. The behavior of `mailx` in saving partial messages is unspecified if the User Portability Utilities Option is not supported and **DEAD** is not defined with the value `/dev/null`.

EDITOR This variable shall be interpreted as a name of a utility to invoke when the `edit` (see 4.40.7.2.8) or `~e` (see 4.40.7.3), command is used. The default is unspecified. This standard specifies the effects of this variable only for systems supporting the User Portability Utilities Option.

HOME This variable shall be interpreted as the pathname of the user's home directory.

LANG This variable shall determine the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. See 2.6.

LC_ALL This variable shall determine the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_CTYPE This variable shall determine the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in arguments and input files) and the handling of case-insensitive address and header-field comparisons.

LC_MESSAGES This variable shall determine the language in which messages should be written.

LC_TIME This variable may determine the format and contents of 7
the date and time strings written by `mailx`. This stan- 7
dard specifies the effects of this variable only for systems 7
supporting the User Portability Utilities Option. 7

LISTER This variable shall be interpreted as a string representing the command for writing the contents of the `folder` directory to standard output when the `folders` command is given (see 4.40.7.2.11). Any string acceptable as a *command_string* operand to the `sh −c` command (see 4.56) shall be valid. If this variable is null or not set, the output command shall be `ls` (see 4.39). The default value
shall be unset. This standard specifies the effects of this 7
variable only for systems supporting the User Portability 7
Utilities Option. 7

MAILRC This variable shall be interpreted as a pathname of the startup file. The default shall be `.mailrc` in the **HOME** directory. The behavior of `mailx` is unspecified if the User Portability Utilities Option is not supported and
MAILRC is not defined with the value `/dev/null`. 7

MBOX This variable shall be interpreted as a pathname of the file to save messages from the system mailbox that have been read. The `exit` command shall override this function, as shall saving the message explicitly in another file. The default shall be `mbox` in the directory named by the
HOME variable. This standard specifies the effects of this 7
variable only for systems supporting the User Portability 7
Utilities Option. 7

PAGER This variable shall be interpreted as a string representing an output filtering and/or pagination command for writing the output to the terminal. Any string acceptable as a *command_string* operand to the `sh −c` command (see 4.56) shall be valid. When standard output is a terminal device, the message output shall be piped through the command if the `mailx` internal variable `crt` is set to a value less the number of lines in the message; see 4.40.7.1. If the **PAGER** variable is null or not set, the paginator shall
be either `more` (see 5.18) or another paginator utility 8
documented in the system documentation. This standard 8
specifies the effects of this variable only for systems sup- 7
porting the User Portability Utilities Option. 7

SHELL The name of a preferred command interpreter. The
default shall be `sh` (see 4.56). This standard specifies the 7
effects of this variable only for systems supporting the 7
User Portability Utilities Option. 7

TERM This variable shall be interpreted as the name of the ter-
minal type, to determine in an unspecified manner, if the 7
internal variable `screen` is not specified, the number of 7
lines in a screenful of headers. If **TERM** is not set or is set 7
to null, an unspecified default terminal type shall be used 7
and the value of a screenful is unspecified. This standard 7
specifies the effects of this variable only for systems sup- 7
porting the User Portability Utilities Option. 7

VISUAL This variable shall be interpreted as a pathname of a utility to invoke when the `visual` command (see 4.40.7.2.38) or ~`v` command-escape (see 4.40.7.3), is used. If this variable is null or not set, the full-screen editor shall be `vi`
(see 5.35). This standard specifies the effects of this vari- 7
able only for systems supporting the User Portability Util- 7
ities Option. 7

4.40.5.4 Asynchronous Events

When `mailx` is in Send Mode and standard input is not a terminal, it shall take the standard action for all signals; see 2.11.5.4.

In Receive Mode, or in Send Mode when standard input is a terminal, if a SIGINT signal is received:

(1) If in command mode, the current command, if there is one, shall be aborted, and a command-mode prompt shall be written.

(2) If in input mode:

  (a) If `ignore` is set, the interrupt shall be ignored, and processing continues.

(b) If the interrupt was received while sending mail, either when in Receive Mode or in Send Mode, a message shall be written, and
another subsequent interrupt, with no other intervening characters 7
typed, shall be required to abort the mail message. If in Receive 7
Mode and another interrupt is received, a command-mode prompt shall be written. If in Send Mode and another interrupt is received, `mailx` shall terminate with a nonzero status.

In both cases listed in (b), if the message is not empty:

[1] If `save` is enabled and the file named by **DEAD** can be created, the message shall be written to the file named by **DEAD** . If the file exists, the message shall be written to replace the contents of the file.

[2] If `save` is not enabled, or the file named by **DEAD** cannot be created, the message shall not be saved.

The `mailx` utility shall take the standard action for all other signals; see 2.11.5.4.

4.40.6 External Effects

4.40.6.1 Standard Output

In command and input modes, all output, including prompts and messages, shall be written to standard output.

4.40.6.2 Standard Error

Used only for diagnostic messages.

4.40.6.3 Output Files

Various `mailx` commands and command escapes can create or add to files, includ-
ing the *mbox* , the dead-letter file, and secondary mailboxes. When `mailx` is used 7
as described in this standard, these files shall be text files, formatted as follows: 7

line beginning with `"From "` 7
[*header-line(s); see 4.40.7.2*] 8
empty line 7
[*body line(s)* 7
empty line] 7
[*line beginning with* `"From "` ...] 8

where each message begins with the `"From "` line shown, preceded by the begin- 8
ning of the file or an empty line. (The `"From "` line is considered to be part of the 8
message header, but not one of the ‘‘header-lines’’ referred to in 4.40.7.2; thus, it 8
is not affected by the `discard`, `ignore`, or `retain` commands). The formats of 8
the remainder of the `"From "` line and any additional header lines are 7
unspecified, except that none shall be empty. The format of a message body line 7

is also unspecified, except that none shall start with `"From "`; `mailx` shall modify any user-entered message body lines beginning with `"From "` by adding one or more characters to precede the `F`.

When a message from the system mailbox or entered by the user is not a text file, it is implementation defined how such a message is stored in files written by `mailx`.

4.40.7 Extended Description

The entire Extended Description subclause (4.40.7 through 4.40.7.3) shall apply only to implementations supporting the User Portability Utilities option.

When `mailx` is invoked using one of the Receive Mode synopsis forms, it shall write the page of header-summary lines (see below) containing the first new message (if `–N` is not specified), or the first unread message if there are no new messages, or the first message if there are no new or unread messages, followed by a prompt indicating `mailx` can accept regular commands (see 4.40.7.2); this is termed *command mode*. When `mailx` is invoked using the Send Mode synopsis and standard input is a terminal, if no subject is specified on the command line and the `asksub` variable is set, a prompt for the subject shall be written. At this point `mailx` is in *input mode*. This input mode is also entered when using one of the Receive Mode synopsis forms and a reply or new message is composed using the `reply`, `Reply`, or `mail` commands. When the message is typed and the end of message is encountered, the message shall be passed to the mail delivery software. Commands can be entered by beginning a line with the escape character [by default, tilde (~)] followed by a single command letter and optional arguments. See 4.40.7.3 for a summary of these commands.

NOTE: For notational convenience, this clause uses the default escape character, tilde, in all references and examples.

At any time, the behavior of `mailx` shall be governed by a set of environmental and internal variables. These are flags and valued parameters that can be set and cleared via the `set` and `unset` commands. See 4.40.5.3 and 4.40.7.1 for summaries of these parameters.

Regular commands are of the form:

[*command* **] [** *msglist* **] [** *argument* ... **]**

If no *command* is specified in command mode, `print` shall be assumed. In input mode, commands shall be recognized by the escape character, and lines not treated as commands shall be taken as input for the message.

In command mode, each message shall be assigned a sequential number, starting with 1.

All messages have a state that affects how they are displayed in the header summary and how they are retained or deleted upon termination of `mailx`. There is at any time the notion of a *current* message, marked by a `>` at the beginning of a line in the header summary. All messages are in one of the following states:

new The message is present in the system mailbox and has not been viewed by the user or moved to any other state. Messages in state *new* when `mailx` quits shall be retained in the system mailbox.

unread The message has been present in the system mailbox for more than one invocation of `mailx` and has not been viewed by the user or moved to any other state. Messages in state *unread* when `mailx` quits shall be retained in the system mailbox.

read The message has been processed by one of the following commands: `~f`, `~m`, `~F`, `~M`, `copy`, `mbox`, `next`, `pipe`, `print`, `Print`, `top`, `type`, `Type`, `undelete`. The `delete`, `dp`, and `dt` commands may also cause the next message to be marked as *read*, depending on the value of the `autoprint` variable. Messages that are in the system mailbox and in state *read* when `mailx` quits shall be
saved in the *mbox*, unless the internal variable `hold` was set. 7
Messages that are in the *mbox* or in a secondary mailbox and in 7
state *read* when `mailx` quits shall be retained in their current location.

deleted The message has been processed by one of the following commands:
`delete`, `dp`, `dt`. A message processed by `save` shall be 7
in state *deleted* unless the internal variable `keepsave` was set. 7
Messages in state *deleted* when `mailx` quits shall be deleted.

preserved The message has been processed by a `preserve` command. When `mailx` quits, the message shall be retained in its current location.

The header-summary line for each message shall indicate the state of the message.

Many commands take an optional list of messages (*msglist*) on which to operate, which defaults to the current message. A *msglist* is a list of message specifications separated by `<blank>`s, which can include:

n Message number *n*.

`+` The next undeleted message, or the next deleted message for the 7
`undelete` command. 7

`-` The next previous undeleted message, or the next previous 7
deleted message for the `undelete` command. 7

`.` The current message.

`^` The first undeleted message, or the first deleted message for the `undelete` command.

`$` The last message.

`*` All messages.

n−*m* An inclusive range of message numbers.

address All messages from *address*; any address as shown in a header summary shall be matchable in this form.

`/`*string* All messages with *string* in the subject line (case ignored).

`:`*c* All messages of type *c*, where *c* shall be one of:

- `d` Deleted messages
- `n` New messages
- `o` Old messages (any not in state *read* or *new*).
- `r` Read messages
- `u` Unread messages

Other commands take an optional message (*message*) on which to operate, which 7
defaults to the current message. All of the forms allowed for *msglist* are also 7
allowed for *message*, but if more than one message is specified, only the first shall 7
be operated on. 7

Other arguments are usually arbitrary strings whose usage depends on the command involved.

At startup time, `mailx` shall take the following steps in sequence:

(1) Establish all variables at their stated default values.

(2) Process command-line options, overriding corresponding default values.

(3) Import any of the **DEAD**, **EDITOR**, **MBOX**, **LISTER**, **PAGER**, **SHELL**, or **VISUAL** variables that are present in the environment, overriding the corresponding default values.

(4) Read `mailx` commands from an unspecified system startup file, unless 7
the `-n` option is given, to initialize any internal `mailx` variables and 7
aliases.

(5) Process the startup file of `mailx` commands named in the user **MAILRC** variable.

Most regular `mailx` commands are valid inside startup files, the most common use being to set up initial display options and alias lists. The following commands shall be invalid in the startup file: `!`, `edit`, `hold`, `mail`, `preserve`, `reply`, `Reply`, `shell`, and `visual`. Any errors in the startup file shall either cause `mailx` to terminate with a diagnostic message and a nonzero status, or to continue after writing a diagnostic message, ignoring the remainder of the lines in the startup file.

A blank line in a startup file shall be ignored. 7

4.40.7.1 `mailx` Internal Variables

The following variables are internal `mailx` variables. Each internal variable can be set via the `mailx set` command at any time. The `unset` and `set no`*name* commands can be used to erase variables.

In the following list, variables shown as

`variable`

represent Boolean values. Variables shown as

`variable=`*value*

shall be assigned string or numeric values. For string values, the rules in 4.40.7.2 concerning file names and quoting also apply.

The defaults specified here may be changed by the implementation-specific system startup file unless the user specifies the `-n` option.

`append` Append messages to the end of the *mbox* file upon termination instead of placing them at the beginning. The default shall be `noappend`. This variable shall not affect the `save` command when saving to the *mbox* .

`ask`
`asksub` Prompt for a subject line on outgoing mail if one is not specified on the command line with the `-s` option. The `ask` and `asksub` forms are synonyms; the system shall refer to `asksub` and `noasksub` in its messages, but shall accept `ask` and `noask` as user input to mean `asksub` and `noasksub`. It shall not be possible to set both `ask` and `noasksub`, or vice versa. The default shall be `asksub`, but no prompting shall be done if standard input is not a terminal.

`askbcc` Prompt for the Bcc list. The default shall be `noaskbcc`.

`askcc` Prompt for the Cc list. The default shall be `noaskcc`.

`autoprint` Enable automatic writing of messages after `delete` and `undelete` commands. The default shall be `noautoprint`.

`bang` Enable the special-case treatment of exclamation-marks (`!`) in escape command lines; see 4.40.7.2.41 and 4.40.7.3. The default shall be `nobang`, disabling the expansion of `!` in the *command* argument to the `~!` command and the `~<!`*command* escape.

`cmd=`*command*
Set the default command to be invoked by the `pipe` command; see 4.40.7.2.21. The default shall be `nocmd`.

`crt=`*number*
Pipe messages having more than *number* lines through the command specified by the value of the **PAGER** variable. The default shall be `nocrt`. If it is set to null, the value used is implementation defined.