NAME

mailx − send and receive Internet mail

SYNOPSIS

mailx [**−BDdEFintv˜**] [**−R** [*address*]] [**−s** *subject*] [**−a** *attachment*] [**−c** *cc-addr*] [**−b** *bcc-addr*] [**−r** *from-addr*] [**−h** *hops*] [**−A** *account*] [**−S** *variable*[=*value*]] *to-addr* . . .
mailx [**−BDdeEHiInNRv˜**] [**−T** *name*] [**−A** *account*] [**−S** *variable*[=*value*]] **−f** [*name*]
mailx [**−BDdeEinNRv˜**] [**−A** *account*] [**−S** *variable*[=*value*]] [**−u** *user*]

DESCRIPTION

Mailx is an intelligent mail processing system, which has a command syntax reminiscent of *ed*(1) with lines replaced by messages. It is based on Berkeley Mail 8.1, is intended to provide the functionality of the POSIX **mailx** command, and offers extensions for MIME, IMAP, POP3, SMTP, and S/MIME. *Mailx* provides enhanced features for interactive use, such as caching and disconnected operation for IMAP, message threading, scoring, and filtering. It is also usable as a mail batch language, both for sending and receiving mail.

The following options are accepted:

−A *name*
Executes an *account* command (see below) for *name* after the startup files have been read.

−a *file* Attach the given file to the message.

−B Make standard input and standard output line-buffered.

−b *address*
Send blind carbon copies to list. List should be a comma-separated list of names.

−c *address*
Send carbon copies to list of users.

−D Start in *disconnected* mode; see the description for the *disconnected* variable option.

−d Enables debugging messages and disables the actual delivery of messages. Unlike *−v*, this option is intended for *mailx* development only.

−e Just check if mail is present in the system mailbox. If yes, return an exit status of zero, else, a non-zero value.

−E If an outgoing message does not contain any text in its first or only message part, do not send it but discard it silently, effectively setting the *skipemptybody* variable at program startup. This is useful for sending messages from scripts started by *cron*(8).

−f [*file*] Read in the contents of the user's mbox (or the specified file) for processing; when *mailx* is quit, it writes undeleted messages back to this file. The string *file* is handled as described for the *folder* command below.

−F Save the message to send in a file named after the local part of the first recipient's address.

−H Print header summaries for all messages and exit.

−h *hops*
Invoke sendmail with the specified hop count. This option has no effect when SMTP is used for sending mail.

−i Ignore tty interrupt signals. This is particularly useful when using *mailx* on noisy phone lines.

−I Shows the 'Newsgroup:' or 'Article-Id:' fields in the header summary. Only applicable in combination with *−f*.

−n Inhibits reading /etc/mail.rc upon startup. This option should be activated for *mailx* scripts that are invoked on more than one machine, because the contents of that file may differ between them.

−N Inhibits the initial display of message headers when reading mail or editing a mail folder.

−q *file* Start the message with the contents of the specified file. May be given in send mode only.

−r *address*
Sets the *From* address. Overrides any *from* variable specified in environment or startup files. Tilde escapes are disabled. The *−r address* options are passed to the mail transfer agent unless SMTP is used. This option exists for compatibility only; it is recommended to set the *from* variable directly instead.

−R | −R *address*
Without any argument any folders will be opened read-only. With argument an reply-to adress is specifed on command line. Only the first argument after the **−R** flag is used as the address.

−s *subject*
Specify subject on command line (only the first argument after the *−s* flag is used as a subject; be careful to quote subjects containing spaces).

−S *variable*[**=***value*]
Sets the internal option *variable* and, in case of a string option, assigns *value* to it.

−T *name*
Writes the 'Message-Id:' and 'Article-Id:' header fields of each message read in the file *name*. Implies *−I*. Compressed files are handled as described for the *folder* command below.

−t The message to be sent is expected to contain a message header with 'To:', 'Cc:', or 'Bcc:' fields giving its recipients. Recipients specified on the command line are ignored.

−u *user* Reads the mailbox of the given user name.

−v Verbose mode. The details of delivery are displayed on the user's terminal.

−V Print *mailx*'s version and exit.

−~ Enable tilde escapes even if not in interactive mode.

Sending mail

To send a message to one or more people, *mailx* can be invoked with arguments which are the names of people to whom the mail will be sent. The user is then expected to type in his message, followed by an 'control-D' at the beginning of a line. The section below Replying to or originating mail, describes some features of *mailx* available to help when composing letters.

Reading mail

In normal usage *mailx* is given no arguments and checks the user's mail out of the post office, then prints out a one line header of each message found. The current message is initially the first message (numbered 1) and can be printed using the print command which can be abbreviated 'p'). The user can move among the messages much as he moves between lines in *ed*(1), with the commands '+' and '−' moving backwards and forwards, and simple numbers.

Disposing of mail

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

Specifying messages

Commands such as print and delete can be given a list of message numbers as arguments to apply to a number of messages at once. Thus '*delete 1 2*' deletes messages 1 and 2, while '*delete 1-5*' deletes messages 1 through 5. In sorted or threaded mode (see the *sort* and *thread* commands), '*delete 1-5*' deletes the messages that are located between (and including) messages 1 through 5 in the sorted/threaded order, as shown in the header summary. The following special message names exist:

:n All new messages.

:o All old messages (any not in state read or new).

:u All unread messages.

:d All deleted messages (for the *undelete* command).

:r All read messages.

:f All ‘flagged’ messages.

:a All answered messages (cf. the *markanswered* variable).

:t All messages marked as draft.

:k All ‘killed’ messages.

:j All messages classified as junk.

. The current message.

; The message that was previously the current message.

, The parent message of the current message, that is the message with the Message-ID given in the ‘In-Reply-To:’ field or the last entry of the ‘References:’ field of the current message.

− The next previous undeleted message, or the next previous deleted message for the *undelete* command. In sorted/threaded mode, the next previous such message in the sorted/threaded order.

+ The next undeleted message, or the next deleted message for the *undelete* command. In sorted/threaded mode, the next such message in the sorted/threaded order.

ˆ The first undeleted message, or the first deleted message for the *undelete* command. In sorted/threaded mode, the first such message in the sorted/threaded order.

$ The last message. In sorted/threaded mode, the last message in the sorted/threaded order.

&*x* In threaded mode, selects the message addressed with *x*, where *x* is any other message specification, and all messages from the thread that begins at it. Otherwise, it is identical to *x*. If *x* is omitted, the thread beginning with the current message is selected.

***** All messages.

‘ All messages that were included in the message list for the previous command.

/*string* All messages that contain *string* in the subject field (case ignored). See also the *searchheaders* variable. If *string* is empty, the string from the previous specification of that type is used again.

address

All messages from *address*. By default, this is a case-sensitive search for the complete email address. If the *allnet* variable is set, only the local part of the addresses is evaluated for the comparison. Otherwise if the *showname* variable is set, a case-sensitive search for the complete real name of a sender is performed. The IMAP-style **(from** *address)* expression can be used instead if substring matches are desired.

(*criterion*)

All messages that satisfy the given IMAP-style SEARCH *criterion*. This addressing mode is available with all types of folders; for folders not located on IMAP servers, or for servers unable to execute the SEARCH command, *mailx* will perform the search locally. Strings must be enclosed by double quotes ‘"’ in their entirety if they contain white space or parentheses; within the quotes, only backslash ‘\’ is recognized as an escape character. All string searches are case-insensitive. When the description indicates that the ‘envelope’ representation of an address field is used, this means that the search string is checked against both a list constructed as

("*real name***" "***source-route***" "***local-part***" "***domain-part***")**

for each address, and the addresses without real names from the respective header field. Criteria can be nested using parentheses.

(*criterion1 criterion2 . . . criterionN*)
All messages that satisfy all of the given criteria.

(or *criterion1 criterion2***)**
All messages that satisfy either *criterion1* or *criterion2*, or both. To connect more than two criteria using ‘or’, (or) specifications have to be nested using additional parentheses, as with ‘(or a (or b c))’; ‘(or a b c)’ means ((a or b) and c). For a simple ‘or’ operation of independent criteria on the lowest nesting level, it is possible to achieve similar effects by using three separate criteria, as with ‘(a) (b) (c)’.

(not *criterion***)**
All messages that do not satisfy *criterion*.

(bcc *string***)**
All messages that contain *string* in the ‘envelope’ representation of the *Bcc:* field.

(cc *string***)**
All messages that contain *string* in the ‘envelope’ representation of the *Cc:* field.

(from *string***)**
All messages that contain *string* in the ‘envelope’ representation of the *From:* field.

(subject *string***)**
All messages that contain *string* in the *Subject:* field.

(to *string***)**
All messages that contain *string* in the ‘envelope’ representation of the *To:* field.

(header *name string***)**
All messages that contain *string* in the specified *Name:* field.

(body *string***)**
All messages that contain *string* in their body.

(text *string***)**
All messages that contain *string* in their header or body.

(larger *size***)**
All messages that are larger than *size* (in bytes).

(smaller *size***)**
All messages that are smaller than *size* (in bytes).

(before *date***)**
All messages that were received before *date*; *date* must be in the form *d*[*d*]**-***mon***-***yyyy*, where *d*[*d*] is the day of the month as one or two digits, *mon* is the name of the month—one of ‘Jan’, ‘Feb’, ‘Mar’, ‘Apr’, ‘May’, ‘Jun’, ‘Jul’, ‘Aug’, ‘Sep’, ‘Oct’, ‘Nov’, or ‘Dec’, and *yyyy* is the year as four digits; e. g. "30-Aug-2004".

(on *date***)**
All messages that were received on the specified date.

(since *date***)**
All messages that were received since the specified date.

(sentbefore *date***)**
All messages that were sent on the specified date.

(senton *date***)**
All messages that were sent on the specified date.

(sentsince *date***)**
All messages that were sent since the specified date.

() The same criterion as for the previous search. This specification cannot be used as part of another criterion. If the previous command line contained more than one independent criterion, the last of those criteria is used.

A practical method to read a set of messages is to issue a *from* command with the search criteria first to check for appropriate messages, and to read each single message then by typing '' repeatedly.

Replying to or originating mail

The *reply* command can be used to set up a response to a message, sending it back to the person who it was from. Text the user types in then, up to an end-of-file, defines the contents of the message. While the user is composing a message, *mailx* treats lines beginning with the character '˜' specially. For instance, typing '˜m' (alone on a line) will place a copy of the current message into the response right shifting it by a tabstop (see *indentprefix* variable, below). Other escapes will set up subject fields, add and delete recipients to the message, attach files to it and allow the user to escape to an editor to revise the message or to a shell to run some commands. (These options are given in the summary below.)

Ending a mail processing session

The user can end a *mailx* session with the quit ('q') command. Messages which have been examined go to the user's mbox file unless they have been deleted in which case they are discarded. Unexamined messages go back to the post office. (See the −f option above).

Personal and systemwide distribution lists

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

alias *cohorts bill ozalp jkf mark kridle@ucbcory*

in the file .mailrc in the user's home directory. The current list of such aliases can be displayed with the alias command in *mailx*. System wide distribution lists can be created by editing /etc/aliases, see *aliases*(5) and *sendmail*(8); these are kept in a different syntax. In mail the user sends, personal aliases will be expanded in mail sent to others so that they will be able to reply to the recipients. System wide aliases are not expanded when the mail is sent, but any reply returned to the machine will have the system wide alias expanded as all mail goes through sendmail.

Recipient address specifications

If the *expandaddr* option is not set (the default), recipient addresses must be names of local mailboxes or Internet mail addresses.

If the *expandaddr* option is set, the following rules apply: When an address is used to name a recipient (in any of To, Cc, or Bcc), names of local mail folders and pipes to external commands can also be specified; the message text is then written to them. The rules are: Any name which starts with a '**|**' character specifies a pipe, the command string following the '**|**' is executed and the message is sent to its standard input; any other name which contains a '**@**' character is treated as a mail address; any other name which starts with a '**+**' character specifies a folder name; any other name which contains a '**/**' character but no '**!**' or '**%**' character before also specifies a folder name; what remains is treated as a mail address. Compressed folders are handled as described for the *folder* command below.

Network mail (Internet / ARPA, UUCP, Berknet)

See *mailaddr*(7) for a description of network addresses. *Mailx* has a number of options which can be set in the .mailrc file to alter its behavior; thus '*set askcc*' enables the askcc feature. (These options are summarized below).

MIME types

For any outgoing attachment, *mailx* tries to determine the content type. It does this by reading MIME type files whose lines have the following syntax:

type/subtype *extension* [*extension* . . .]

where type/subtype are strings describing the file contents, and extension is the part of a filename starting after the last dot. Any line not immediately beginning with an ASCII alphabetical character is ignored by *mailx*. If there is a match with the extension of the file to attach, the given type/subtype pair is used.

Otherwise, or if the filename has no extension, the content types text/plain or application/octet-stream are used, the first for text or international text files, the second for any file that contains formatting characters other than newlines and horizontal tabulators.

Character sets

Mailx normally detects the character set of the terminal using the LC_CTYPE locale setting. If the locale cannot be used appropriately, the *ttycharset* variable should be set to provide an explicit value. When reading messages, their text is converted to the terminal character set if possible. Unprintable characters and illegal byte sequences are detected and replaced by Unicode substitute characters or question marks unless the *print-all-chars* is set at initialization time.

The character set for outgoing messages is not necessarily the same as the one used on the terminal. If an outgoing text message contains characters not representable in US-ASCII, the character set being used must be declared within its header. Permissible values can be declared using the *sendcharsets* variable, separated by commas; *mailx* tries each of the values in order and uses the first appropriate one. If the message contains characters that cannot be represented in any of the given character sets, the message will not be sent, and its text will be saved to the 'dead.letter' file. Messages that contain NUL bytes are not converted.

Outgoing attachments are converted if they are plain text. If the *sendcharsets* variable contains more than one character set name, the ~@ tilde escape will ask for the character sets for individual attachments if it is invoked without arguments.

Best results are usually achieved when *mailx* is run in a UTF-8 locale on a UTF-8 capable terminal. In this setup, characters from various countries can be displayed, while it is still possible to use more simple character sets for sending to retain maximum compatibility with older mail clients.

Commands

Each command is typed on a line by itself, and may take arguments following the command word. The command need not be typed in its entirety – the first command which matches the typed prefix is used. For commands which take message lists as arguments, if no message list is given, then the next message forward which satisfies the command's requirements is used. If there are no messages forward of the current message, the search proceeds backwards, and if there are no good messages at all, *mailx* types '*applicable messages*' and aborts the command. If the command begins with a # sign, the line is ignored.

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 are treated literally as part of the argument. A double-quote will be treated literally within single-quotes and vice versa. These special properties of the quote marks occur only when they are paired at the beginning and end of the argument.
- A backslash outside of the enclosing quotes is discarded and the following character is treated literally as part of the argument.
- An unquoted backslash at the end of a command line is discarded and the next line continues the command.

Filenames, where expected, are subjected to the following transformations, in sequence:

- If the filename begins with an unquoted plus sign, and the *folder* variable is defined, the plus sign will be replaced by the value of the *folder* variable followed by a slash. If the *folder* variable is unset or is set to null, the filename will be unchanged.
- Shell word expansions are applied to the filename. If more than a single pathname results from this expansion and the command is expecting one file, an error results.

The following commands are provided:

– Print out the preceding message. If given a numeric argument n, goes to the n'th previous message and prints it.

? Prints a brief summary of commands.

! Executes the shell (see *sh*(1) and *csh*(1)) command which follows.

| A synonym for the *pipe* command.

account
(ac) Creates, selects or lists an email account. An account is formed by a group of commands, primarily of those to set variables. With two arguments, of which the second is a '{', the first argument gives an account name, and the following lines create a group of commands for that account until a line containing a single '}' appears. With one argument, the previously created group of commands for the account name is executed, and a *folder* command is executed for the system mailbox or inbox of that account. Without arguments, the list of accounts and their contents are printed. As an example,

```
account myisp {
    set folder=imaps://mylogin@imap.myisp.example
    set record=+Sent
    set from="myname@myisp.example (My Name)"
    set smtp=smtp.myisp.example
}
```

creates an account named 'myisp' which can later be selected by specifying 'account myisp'.

alias (a) With no arguments, prints out all currently-defined aliases. With one argument, prints out that alias. With more than one argument, creates a new alias or changes an old one.

alternates
(alt) The alternates command is useful if the user has accounts on several machines. It can be used to inform *mailx* that the listed addresses all belong to the invoking user. When he replies to messages, *mailx* will not send a copy of the message to any of the addresses listed on the alternates list. If the alternates command is given with no argument, the current set of alternate names is displayed.

answered
(ans) Takes a message list and marks each message as a having been answered. This mark has no technical meaning in the mail system; it just causes messages to be marked in the header summary, and makes them specially addressable.

cache Only applicable to cached IMAP mailboxes; takes a message list and reads the specified messages into the IMAP cache.

call Calls a macro (see the *define* command).

cd Same as chdir.

certsave
Only applicable to S/MIME signed messages. Takes a message list and a file name and saves the certificates contained within the message signatures to the named file in both human-readable and PEM format. The certificates can later be used to send encrypted messages to the messages' originators by setting the *smime-encrypt-user@host* variable.

chdir (ch) Changes the user's working directory to that specified, if given. If no directory is given, then changes to the user's login directory.

classify
(cl) Takes a list of messages and examines their contents for characteristics of junk mail using Bayesian filtering. Messages considered to be junk are then marked as such. The junk mail database is not changed.

collapse
(coll) Only applicable to threaded mode. Takes a message list and makes all replies to these messages invisible in header summaries, unless they are in state 'new'.

connect
(conn) If operating in disconnected mode on an IMAP mailbox, switch to online mode and connect to the mail server while retaining the mailbox status. See the description of the *disconnected* variable for more information.

copy (c) 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 the user quits. Compressed files and IMAP mailboxes are handled as described for the *folder* command.

Copy (C) Similar to *copy*, but saves the messages in a file named after the local part of the sender address of the first message.

decrypt
(dec) For unencrypted messages, this command is identical to *copy*. Encrypted messages are first decrypted, if possible, and then copied.

Decrypt
(Dec) Similar to *decrypt*, but saves the messages in a file named after the local part of the sender address of the first message.

define (def) Defines a macro. A macro definition is a sequence of commands in the following form:

define *name* **{**
command1
command2
. . .
commandN
}

Once defined, a macro can be explicitly invoked using the *call* command, or can be implicitly invoked by setting the *folder-hook* or *folder-hook-fullname* variables.

defines Prints the currently defined macros including their contents.

delete (d) Takes a list of messages as argument and marks them all as deleted. Deleted messages will not be saved in mbox, nor will they be available for most other commands.

discard
Same as ignore.

disconnect
(disco) If operating in online mode on an IMAP mailbox, switch to disconnected mode while retaining the mailbox status. See the description of the *disconnected* variable for more information. A list of messages may optionally be given as argument; the respective messages are then read into the cache before the connection is closed. Thus 'disco *' makes the entire current mailbox available for disconnected use.

dp or **dt**
Deletes the current message and prints the next message. If there is no next message, *mailx* says '*at EOF*'.

draft Takes a message list and marks each message as a draft. This mark has no technical meaning in the mail system; it just causes messages to be marked in the header summary, and makes them specially addressable.

echo Echoes its arguments, resolving special names as documented for the folder command. The escape sequences '**\a**', '**\b**', '**\c**', '**\f**', '**\n**', '**\r**', '**\t**', '**\v**', '****', and '**\0***num*' are interpreted as with the *echo*(1) command.

edit (e) Takes a list of messages and points the text editor at each one in turn. Modified contents are discarded unless the *writebackedited* variable is set.

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 (ex or x) Effects an immediate return to the Shell without modifying the user's system mailbox, his mbox file, or his edit file in −f.

file (fi) The same as folder.

flag (fl) Takes a message list and marks the messages as 'flagged' for urgent/special attention. This mark has no technical meaning in the mail system; it just causes messages to be highlighted in the header summary, and makes them specially addressable.

folders With no arguments, list the names of the folders in the folder directory. With an existing folder as an argument, lists then names of folders below the named folder; e. g. the command 'folders @' lists the folders on the base level of the current IMAP server. See also the *imap-list-depth* variable.

folder (fold) The folder command switches to a new mail file or folder. With no arguments, it tells the user which file he is currently reading. If an argument is given, it will write out changes (such as deletions) the user has made in the current file and read in the new file. Some special conventions are recognized for the name. **#** means the previous file, **%** means the invoking user's system mailbox, **%***user* means *user's* system mailbox, **&** means the invoking user's mbox file, and **+***file means a file* in the folder directory. **%:***filespec* expands to the same value as *filespec*, but the file is handled as a system mailbox e. g. by the mbox and save commands. If the name matches one of the strings defined with the *shortcut* command, it is replaced by its long form and expanded. If the name ends with **.gz** or **.bz2**, it is treated as compressed with *gzip*(1) or *bzip2*(1), respectively. Likewise, if *name* does not exist, but either *name***.gz** or *name***.bz2** exists, the compressed file is used. If *name* refers to a directory with the subdirectories 'tmp', 'new', and 'cur', it is treated as a folder in *maildir* format. A name of the form

> *protocol***://**[*user***@**]*host*[**:***port*][**/***file*]

is taken as an Internet mailbox specification. The supported protocols are currently **imap** (IMAP v4r1), **imaps** (IMAP with SSL/TLS encryption), **pop3** (POP3), and **pop3s** (POP3 with SSL/TLS encryption). If *user* contains special characters, in particular '/' or '%', they must be escaped in URL notation, as '%2F' or '%25'. The optional *file* part applies to IMAP only; if it is omitted, the default 'INBOX' is used. If *mailx* is connected to an IMAP server, a name of the form @*mailbox* refers to the *mailbox* on that server. If the 'folder' variable refers to an IMAP account, the special name '%' selects the 'INBOX' on that account.

Followup
(F) Similar to *Respond*, but saves the message in a file named after the local part of the first recipient's address.

followup
(fo) Similar to *respond*, but saves the message in a file named after the local part of the first recipient's address.

followupall
Similar to *followup*, but responds to all recipients regardless of the *flipr* and *Replyall* variables.

followupsender
Similar to *Followup*, but responds to the sender only regardless of the *flipr* and *Replyall* variables.

forward
(fwd) Takes a message and the address of a recipient and forwards the message to him. The text of the original message is included in the new one, with the value of the *fwdheading* variable printed

before. The *fwdignore* and *fwdretain* commands specify which header fields are included in the new message. Only the first part of a multipart message is included unless the *forward-as-attachment* option is set.

Forward
(Fwd) Similar to *forward*, but saves the message in a file named after the local part of the recipient's address.

from (f) Takes a list of messages and prints their message headers, piped through the pager if the output does not fit on the screen.

fwdignore
Specifies which header fields are to be ignored with the *forward* command. This command has no effect when the *forward-as-attachment* option is set.

fwdretain
Specifies which header fields are to be retained with the *forward* command. *fwdretain* overrides *fwdignore*. This command has no effect when the *forward-as-attachment* option is set.

good (go) Takes a list of messages and marks all of them as not being junk mail. Data from these messages is then inserted into the junk mail database for future classification.

headers
(h) Lists the current range of headers, which is an 18-message group. If a '+' argument is given, then the next 18-message group is printed, and if a '−' argument is given, the previous 18-message group is printed.

help A synonym for ?.

hold (ho, also preserve) Takes a message list and marks each message therein to be saved in the user's system mailbox instead of in mbox. Does not override the delete command. *mailx* deviates from the POSIX standard with this command, as a 'next' command issued after 'hold' will display the following message, not the current one.

if Commands in *mailx*'s startup files can be executed conditionally depending on whether the user is sending or receiving mail with the if command. For example:

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

An else form is also available:

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

Note that the only allowed conditions are **receive**, **send**, and **term** (execute command if standard input is a tty).

ignore Add the list of header fields named to the ignored list. Header fields in the ignore list are not printed on the terminal when a message is printed. This command is very handy for suppression of certain machine-generated header fields. 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.

imap Sends command strings directly to the current IMAP server. *Mailx* operates always in IMAP *selected state* on the current mailbox; commands that change this will produce undesirable results and should be avoided. Useful IMAP commands are:

create Takes the name of an IMAP mailbox as an argument and creates it.

getquotaroot
Takes the name of an IMAP mailbox as an argument and prints the quotas that apply to the mailbox. Not all IMAP servers support this command.

namespace
Takes no arguments and prints the Personal Namespaces, the Other User's Namespaces, and the Shared Namespaces. Each namespace type is printed in parentheses; if there are multiple namespaces of the same type, inner parentheses separate them. For each namespace, a namespace prefix and a hierarchy separator is listed. Not all IMAP servers support this command.

inc Same as *newmail*.

junk (j) Takes a list of messages and marks all of them as junk mail. Data from these messages is then inserted into the junk mail database for future classification.

kill (k) Takes a list of messages and 'kills' them. Killed messages are not printed in header summaries, and are ignored by the *next* command. The *kill* command also sets the score of the messages to negative infinity, so that subsequent *score* commands will not unkill them again. Killing is only effective for the current session on a folder; when it is quit, all messages are automatically unkilled.

list Prints the names of all available commands.

Mail (M) Similar to *mail*, but saves the message in a file named after the local part of the first recipient's address.

mail (m) Takes as argument login names and distribution group names and sends mail to those people.

mbox Indicate that a list of messages be sent to mbox in the user's home directory when *mailx* is quit. This is the default action for messages if unless the *hold* option is set. *mailx* deviates from the POSIX standard with this command, as a 'next' command issued after 'mbox' will display the following message, not the current one.

move (mv) Acts like *copy*, but marks the messages for deletion if they were transferred successfully.

Move (Mv) Similar to *move*, but moves the messages to a file named after the local part of the sender address of the first message.

newmail
Checks for new mail in the current folder without committing any changes before. If new mail is present, a message is printed. If the *header* variable is set, the headers of each new message are also printed.

next (n) like + or CR) Goes to the next message in sequence and types it. With an argument list, types the next matching message.

New Same as *unread*.

new Same as *unread*.

online Same as *connect*.

noop If the current folder is located on an IMAP or POP3 server, a NOOP command is sent. Otherwise, no operation is performed.

Pipe (Pi) Like *pipe* but also pipes ignored header fields and all parts of MIME *multipart/alternative* messages.

pipe (pi) Takes a message list and a shell command and pipes the messages through the command. Without an argument, the current message is piped through the command given by the *cmd* variable. If the *page* variable is set, every message is followed by a formfeed character.

preserve
(pre) A synonym for *hold*.

Print (P) Like *print* but also prints out ignored header fields and all parts of MIME *multipart/alternative* messages. See also *print*, *ignore*, and *retain*.

print (p) Takes a message list and types out each message on the user's terminal. If the message is a MIME multipart message, all parts with a content type of 'text' or 'message' are shown, the other are hidden except for their headers. Messages are decrypted and converted to the terminal character set if necessary.

probability
(prob) For each word given as argument, the contents of its junk mail database entry are printed.

quit (q) Terminates the session, saving all undeleted, unsaved messages in the user's mbox file in his login directory, preserving all messages marked with hold or preserve or never referenced in his system mailbox, and removing all other messages from his system mailbox. If new mail has arrived during the session, the message '*You have new mail*' is given. If given while editing a mailbox file with the −f flag, then the edit file is rewritten. A return to the Shell is effected, unless the rewrite of edit file fails, in which case the user can escape with the exit command.

redirect
(red) Same as *resend*.

Redirect
(Red) Same as *Resend*.

remove
(rem) Removes the named folders. The user is asked for confirmation in interactive mode.

rename
(ren) Takes the name of an existing folder and the name for the new folder and renames the first to the second one. Both folders must be of the same type and must be located on the current server for IMAP.

Reply (R) Reply to originator. Does not reply to other recipients of the original message.

reply (r) Takes a message list and sends mail to the sender and all recipients of the specified message. The default message must not be deleted.

replyall
Similar to *reply*, but responds to all recipients regardless of the *flipr* and *Replyall* variables.

replysender
Similar to *Reply*, but responds to the sender only regardless of the *flipr* and *Replyall* variables.

Resend
Like *resend*, but does not add any header lines. This is not a way to hide the sender's identity, but useful for sending a message again to the same recipients.

resend Takes a list of messages and a user name and sends each message to the named user. 'Resent-From:' and related header fields are prepended to the new copy of the message.

Respond
Same as *Reply*.

respond
Same as *reply*.

respondall
Same as *replyall*.

respondsender
Same as *replysender*.

retain Add the list of header fields named to the retained list. Only the header fields in the retain list are shown on the terminal when a message is printed. 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 (S) Similar to *save*, but saves the messages in a file named after the local part of the sender of the first message instead of taking a filename argument.

save (s) Takes a message list and a filename and appends each message in turn to the end of the file. If no filename is given, the mbox file is used. The filename in quotes, followed by the line count and character count is echoed on the user's terminal. If editing a system mailbox, the messages are marked for deletion. Compressed files and IMAP mailboxes are handled as described for the −*f* command line option above.

savediscard
Same as saveignore.

saveignore
Saveignore is to save what ignore is to print and type. Header fields thus marked are filtered out when saving a message by save or when automatically saving to mbox. This command should only be applied to header fields that do not contain information needed to decode the message, as MIME content fields do. If saving messages on an IMAP account, ignoring fields makes it impossible to copy the data directly on the server, thus operation usually becomes much slower.

saveretain
Saveretain is to save what retain is to print and type. Header fields thus marked are the only ones saved with a message when saving by save or when automatically saving to mbox. Saveretain overrides saveignore. The use of this command is strongly discouraged since it may strip header fields that are needed to decode the message correctly.

score (sc) Takes a message list and a floating point number and adds the number to the score of each given message. All messages start at score 0 when a folder is opened. When the score of a message becomes negative, it is 'killed' with the effects described for the *kill* command; otherwise if it was negative before and becomes positive, it is 'unkilled'. Scores only refer to the currently opened instance of a folder.

set (se) With no arguments, prints all variable values, piped through the pager if the output does not fit on the screen. Otherwise, sets option. Arguments are of the form option=value (no space before or after =) or option. Quotation marks may be placed around any part of the assignment statement to quote blanks or tabs, i. e. '*set indentprefix="−>"*'. If an argument begins with **no**, as in '**set no***save*', the effect is the same as invoking the *unset* command with the remaining part of the variable ('*unset save*').

seen Takes a message list and marks all messages as having been read.

shell (sh) Invokes an interactive version of the shell.

shortcut
Defines a shortcut name and its string for expansion, as described for the *folder* command. With no arguments, a list of defined shortcuts is printed.

show (Sh) Like *print*, but performs neither MIME decoding nor decryption so that the raw message text is shown.

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

sort Create a sorted representation of the current folder, and change the *next* command and the addressing modes such that they refer to messages in the sorted order. Message numbers are the same as in regular mode. If the *header* variable is set, a header summary in the new order is also printed. Possible sorting criteria are:

date Sort the messages by their 'Date:' field, that is by the time they were sent.

from Sort messages by the value of their 'From:' field, that is by the address of the sender. If the *showname* variable is set, the sender's real name (if any) is used.

size Sort the messages by their size.

score Sort the messages by their score.

status Sort the messages by their message status (new, read, old, etc.).

subject Sort the messages by their subject.

thread Create a threaded order, as with the *thread* command.

to Sort messages by the value of their 'To:' field, that is by the address of the recipient. If the *showname* variable is set, the recipient's real name (if any) is used.

If no argument is given, the current sorting criterion is printed.

source The source command reads commands from a file.

thread (th) Create a threaded representation of the current folder, i. e. indent messages that are replies to other messages in the header display, and change the *next* command and the addressing modes such that they refer to messages in the threaded order. Message numbers are the same as in unthreaded mode. If the *header* variable is set, a header summary in threaded order is also printed.

top Takes a message list and prints the top few lines of each. The number of lines printed is controlled by the variable toplines and defaults to five.

touch Takes a message list and marks the messages for saving in the *mbox* file. *mailx* deviates from the POSIX standard with this command, as a 'next' command issued after 'mbox' will display the following message, not the current one.

Type (T) Identical to the Print command.

type (t) A synonym for print.

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

unanswered
Takes a message list and marks each message as not having been answered.

uncollapse
(unc) Only applicable to threaded mode. Takes a message list and makes the message and all replies to it visible in header summaries again. When a message becomes the current message, it is automatically made visible. Also when a message with collapsed replies is printed, all of these are automatically uncollapsed.

undef Undefines each of the named macros. It is not an error to use a name that does not belong to one of the currently defined macros.

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

undraft
Takes a message list and marks each message as a draft.

unflag Takes a message list and marks each message as not being 'flagged'.

unfwdignore
Removes the header field names from the list of ignored fields for the *forward* command.

unfwdretain
Removes the header field names from the list of retained fields for the *forward* command.

ungood
Takes a message list and undoes the effect of a *good* command that was previously applied on exactly these messages.

unignore
Removes the header field names from the list of ignored fields.

unjunk
Takes a message list and undoes the effect of a *junk* command that was previously applied on exactly these messages.

unkill Takes a message list and 'unkills' each message. Also sets the score of the messages to 0.

Unread
Same as *unread*.

unread (U) Takes a message list and marks each message as not having been read.

unretain
Removes the header field names from the list of retained fields.

unsaveignore
Removes the header field names from the list of ignored fields for saving.

unsaveretain
Removes the header field names from the list of retained fields for saving.

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

unshortcut
Deletes the shortcut names given as arguments.

unsort Disable sorted or threaded mode (see the *sort* and *thread* commands), return to normal message order and, if the *header* variable is set, print a header summary.

unthread
(unth) Same as *unsort*.

verify (verif) Takes a message list and verifies each message. If a message is not an S/MIME signed message, verification will fail for it. The verification process checks if the message was signed using a valid certificate, if the message sender's email address matches one of those contained within the certificate, and if the message content has been altered.

visual (v) Takes a message list and invokes the display editor on each message. Modified contents are discarded unless the *writebackedited* variable is set.

write (w) For conventional messages, the body without all headers is written. The output is decrypted and converted to its native format, if necessary. If the output file exists, the text is appended.—If a message is in MIME multipart format, its first part is written to the specified file as for conventional messages, and the user is asked for a filename to save each other part; if the contents of the first part are not to be saved, 'write /dev/null' can be used. For the second and subsequent parts, if the filename given starts with a '|' character, the part is piped through the remainder of the filename interpreted as a shell command. In non-interactive mode, only the parts of the multipart message that have a filename given in the part header are written, the other are discarded. The original message is never marked for deletion in the originating mail folder. For attachments, the contents of the destination file are overwritten if the file previously existed. No special handling of compressed files is performed.

xit (x) A synonym for exit.

z *Mailx* presents message headers in windowfuls as described under the headers command. The z command scrolls to the next window of messages. If an argument is given, it specifies the window to use. A number prefixed by '+' or '−' indicates that the window is calculated in relation to the current position. A number without a prefix specifies an absolute window number, and a '$' lets *mailx* scroll to the last window of messages.

Z Similar to *z*, but scrolls to the next or previous window that contains at least one new or 'flagged' message.

Tilde escapes

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

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

~. Same effect as typing the end-of-file character.

~< *filename*
Identical to ~r.

~<!*command*
Command is executed using the shell. Its standard output is inserted into the message.

~@ [*filename* . . .]
With no arguments, edit the attachment list. First, the user can edit all existing attachment data. If an attachment's file name is left empty, that attachment is deleted from the list. When the end of the attachment list is reached, *mailx* will ask for further attachments, until an empty file name is given. If *filename* arguments are specified, all of them are appended to the end of the attachment list. Filenames which contain white space can only be specified with the first method (no *filename* arguments).

~A Inserts the string contained in the *Sign* variable (same as '~i Sign'). The escape sequences '**\t**' (tabulator) and '**\n**' (newline) are understood.

~a Inserts the string contained in the **sign** variable (same as '~i sign'). The escape sequences '**\t**' (tabulator) and '**\n**' (newline) are understood.

~b*name* . . .
Add the given names to the list of carbon copy recipients but do not make the names visible in the Cc: line ('blind' carbon copy).

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

~d Read the file 'dead.letter' from the user's home directory into the message.

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

~f*messages*
Read the named messages into the message being sent. If no messages are specified, read in the current message. Message headers currently being ignored (by the ignore or retain command) are not included. For MIME multipart messages, only the first printable part is included.

~F*messages*
Identical to ~f, except all message headers and all MIME parts are included.

~h Edit the message header fields 'To:', 'Cc:', 'Bcc:', and 'Subject:' by typing each one in turn and allowing the user to append text to the end or modify the field by using the current terminal erase and kill characters.

~H Edit the message header fields 'From:', 'Reply-To:', 'Sender:', and 'Organization:' in the same manner as described for *~h*. The default values for these fields originate from the *from*, *replyto*, and *ORGANIZATION* variables. If this tilde command has been used, changing the variables has no effect on the current message anymore.

~i*variable*
Insert the value of the specified variable into the message adding a newline character at the end. If the variable is unset or empty, the message remains unaltered. The escape sequences '**\t**' (tabulator) and '**\n**' (newline) are understood.

~m*messages*
Read the named messages into the message being sent, indented by a tab or by the value of indentprefix. If no messages are specified, read the current message. Message headers currently being ignored (by the ignore or retain command) are not included. For MIME multipart messages, only the first printable part is included.

~M*messages*
Identical to ~m, except all message headers and all MIME parts are included.

~p Print out the message collected so far, prefaced by the message header fields and followed by the attachment list, if any. If the message text is longer than the screen size, it is piped through the pager.

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

~R*string*
Use string as the Reply-To field.

~r *filename*
Read the named file into the message.

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

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

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

~w *filename*
Write the message onto the named file. If the file exists, the message is appended to it.

~x Same as ~q, except that the message is not saved to the 'dead.letter' file.

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

~:*mailx-command*
Execute the given *mailx* command. Not all commands, however, are allowed.

~_*mailx-command*
Identical to ~:.

~~*string* Insert the string of text in the message prefaced by a single ~. If the escape character has been changed, that character must be doubled in order to send it at the beginning of a line.

Variable options

Options are controlled via set and unset commands, see their entries for a syntax description. An option is also set if it is passed to *mailx* as part of the environment (this is not restricted to specific variables as in the POSIX standard). A value given in a startup file overrides a value imported from the environment. Options may be either binary, in which case it is only significant to see whether they are set or not; or string, in which case the actual value is of interest.

Binary options

The binary options include the following:

allnet Causes only the local part to be evaluated when comparing addresses.

append
Causes messages saved in mbox to be appended to the end rather than prepended. This should always be set.

ask or **asksub**
Causes *mailx* to prompt for the subject of each message sent. If the user responds with simply a newline, no subject field will be sent.

askatend
Causes the prompts for 'Cc:' and 'Bcc:' lists to appear after the message has been edited.

askattach
If set, *mailx* asks for files to attach at the end of each message. Responding with a newline indicates not to include an attachment.

askcc
Causes the user to be prompted for additional carbon copy recipients (at the end of each message if *askatend* or *bsdcompat* is set). Responding with a newline indicates the user's satisfaction with the current list.

askbcc
Causes the user to be prompted for additional blind carbon copy recipients (at the end of each message if *askatend* or *bsdcompat* is set). Responding with a newline indicates the user's satisfaction with the current list.

asksign
Causes the user to be prompted if the message is to be signed at the end of each message. The *smime-sign* variable is ignored when this variable is set.

autocollapse
Causes threads to be collapsed automatically when threaded mode is entered (see the *collapse* command).

autoinc
Same as *newmail*.

autoprint
Causes the delete command to behave like dp – thus, after deleting a message, the next one will be typed automatically.

autothread
Causes threaded mode (see the *thread* command) to be entered automatically when a folder is opened.

bang
Enables the substitution of '**!**' by the contents of the last command line in shell escapes.

bsdannounce
Causes automatic display of a header summary after executing a *folder* command.

bsdcompat
Sets some cosmetical features to traditional BSD style; has the same affect as setting 'askatend' and all other variables prefixed with 'bsd', setting prompt to '& ', and changing the default pager to *more*.

bsdflags
Changes the letters printed in the first column of a header summary to traditional BSD style.

bsdheadline
Changes the display of columns in a header summary to traditional BSD style.

bsdmsgs
Changes some informational messages to traditional BSD style.

bsdorder
Causes the 'Subject:' field to appear immediately after the 'To:' field in message headers and with the *~h* tilde command.

bsdset
Changes the output format of the *set* command to traditional BSD style.

chained-junk-tokens
Normally, the Bayesian junk mail filter bases its classifications on single word tokens extracted from messages. If this option is set, adjacent words are combined to pairs, which are then used as additional tokens. This usually improves the accuracy of the filter, but also increases the junk mail database five- to tenfold.

datefield
The date in a header summary is normally the date of the mailbox 'From ' line of the message. If this variable is set, the date as given in the 'Date:' header field is used, converted to local time.

debug Prints debugging messages and disables the actual delivery of messages. Unlike *verbose*, this option is intended for *mailx* development only.

disconnected
When an IMAP mailbox is selected and this variable is set, no connection to the server is initiated. Instead, data is obtained from the local cache (see *imap-cache*). Mailboxes that are not present in the cache and messages that have not yet entirely been fetched from the server are not available; to fetch all messages in a mailbox at once, the command 'copy * /dev/null' can be used while still in *online* mode. Changes that are made to IMAP mailboxes in disconnected mode are queued and committed later when a connection to that server is opened in online mode. This procedure is not completely reliable since it cannot be guaranteed that the IMAP unique identifiers (UIDs) on the server still match the ones in the cache at that time. Data is saved to 'dead.letter' when this problem occurs.

disconnected-*user@host*
The specified account is handled as described for the *disconnected* variable above, but other accounts are not affected.

dot The binary option dot causes *mailx* to interpret a period alone on a line as the terminator of a message the user is sending.

editheaders
When a message is edited while being composed, its header is included in the editable text. 'To:', 'Cc:', 'Bcc:', 'Subject:', 'From:', 'Reply-To:', 'Sender:', and 'Organization:' fields are accepted within the header, other fields are ignored.

emptybox
If set, an empty mailbox file is not removed. This may improve the interoperability with other mail user agents when using a common folder directory.

emptystart
If the mailbox is empty, *mailx* normally prints *'No mail for user'* and exits immediately. If this option is set, *mailx* starts even with an empty mailbox.

expandaddr
Causes *mailx* to expand message recipient addresses, as explained in the section, Recipient address specifications.

flipr Exchanges the *Respond* with the *respond* commands and vice-versa.

forward-as-attachment
Original messages are normally sent as inline text with the *forward* command, and only the first part of a multipart message is included. With this option, messages are sent as MIME *message/rfc822* attachments, and all of their parts are included. The *fwdignore* and *fwdretain* options are ignored when the *forward-as-attachment* option is set.

fullnames
When replying to a message, *mailx* normally removes the comment parts of email addresses, which by convention contain the full names of the recipients. If this variable is set, such stripping is not performed, and comments are retained.

header Causes the header summary to be written at startup and after commands that affect the number of messages or the order of messages in the current folder; enabled by default.

hold This option is used to hold messages in the system mailbox by default.

ignore Causes interrupt signals from the terminal to be ignored and echoed as @'s.

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

imap-use-starttls
Causes *mailx* to issue a STARTTLS command to make an unencrypted IMAP session SSL/TLS encrypted. This functionality is not supported by all servers, and is not used if the session is already encrypted by the IMAPS method.

imap-use-starttls-*user@host*
Activates *imap-use-starttls* for a specific account.

keep This option causes *mailx* to truncate the user's system mailbox instead of deleting it when it is empty. This should always be set, since it prevents malicious users from creating fake mail folders in a world-writable spool directory.

keepsave
When a message is saved, it is usually discarded from the originating folder when *mailx* is quit. Setting this option causes all saved message to be retained.

markanswered
When a message is replied to and this variable is set, it is marked as having been answered. This mark has no technical meaning in the mail system; it just causes messages to be marked in the header summary, and makes them specially addressable.

metoo Usually, when a group is expanded that contains the sender, the sender is removed from the expansion. Setting this option causes the sender to be included in the group.

newmail
Checks for new mail in the current folder each time the prompt is printed. For IMAP mailboxes, the server is then polled for new mail, which may result in delayed operation if the connection to the server is slow. A *maildir* folder must be re-scanned to determine if new mail has arrived.

If this variable is set to the special value **nopoll**, an IMAP server is not actively asked for new mail, but new mail may still be detected and announced with any other IMAP command that is sent to the server. A *maildir* folder is not scanned then.

In any case, the IMAP server may send notifications about messages that have been deleted on the server by another process or client. In this case, 'Expunged *n* messages' is printed regardless of this variable, and message numbers may have changed.

noheader
Setting the option noheader is the same as giving the −N flag on the command line.

outfolder
Causes the filename given in the *record* variable and the sender-based filenames for the *Copy* and *Save* commands to be interpreted relative to the directory given in the *folder* variable rather than to the current directory unless it is an absolute pathname.

page If set, each message the *pipe* command prints out is followed by a formfeed character.

piperaw
Send messages to the *pipe* command without performing MIME and character set conversions.

pop3-use-apop
If this variable is set, the APOP authentication method is used when a connection to a POP3 server is initiated. The advantage of this method over the usual USER/PASS authentication is that the password is not sent over the network in clear text. The connection fails if the server does not

support the APOP command.

pop3-use-apop-*user@host*
Enables *pop3-use-apop* for a specific account.

pop3-use-starttls
Causes *mailx* to issue a STLS command to make an unencrypted POP3 session SSL/TLS encrypted. This functionality is not supported by all servers, and is not used if the session is already encrypted by the POP3S method.

pop3-use-starttls-*user@host*
Activates *pop3-use-starttls* for a specific account.

print-all-chars
This option causes all characters to be considered printable. It is only effective if given in a startup file. With this option set, some character sequences in messages may put the user's terminal in an undefined state when printed; it should only be used as a last resort if no working system locale can be found.

print-alternatives
When a MIME message part of type *multipart/alternative* is displayed and it contains a subpart of type *text/plain*, other parts are normally discarded. Setting this variable causes all subparts to be displayed, just as if the surrounding part was of type *multipart/mixed*.

quiet Suppresses the printing of the version when first invoked.

record-resent
If both this variable and the *record* variable are set, the *resend* and *Resend* commands save messages to the *record* folder as it is normally only done for newly composed messages.

reply-in-same-charset
If this variable is set, *mailx* first tries to use the same character set of the original message for replies. If this fails, the *sendcharsets* variable is evaluated as usual.

Replyall
Reverses the sense of reply and Reply commands.

save When the user aborts a message with two RUBOUT (interrupt characters) *mailx* copies the partial letter to the file 'dead.letter' in the home directory. This option is set by default.

searchheaders
If this option is set, then a message-list specifier in the form '*/x:y*' will expand to all messages containing the substring '*y*' in the header field '*x*'. The string search is case insensitive.

sendwait
When sending a message, wait until the mail transfer agent exits before accepting further commands. If the mail transfer agent returns a non-zero exit status, the exit status of mailx will also be non-zero.

showlast
Setting this option causes *mailx* to start at the last message instead of the first one when opening a mail folder.

showname
Causes *mailx* to use the sender's real name instead of the plain address in the header field summary and in message specifications.

showto Causes the recipient of the message to be shown in the header summary if the message was sent by the user.

skipemptybody
If an outgoing message does not contain any text in its first or only message part, do not send it but discard it silently (see also the −*E* option).

smime-force-encryption
Causes *mailx* to refuse sending unencrypted messages.

smime-sign
If this variable is set, outgoing messages are S/MIME signed with the user's private key. Signing a message enables a recipient to verify that the sender used a valid certificate, that the email addresses in the certificate match those in the message header, and that the message content has not been altered. It does not change the message text, and people will be able to read the message as usual.

smime-no-default-ca
Do not load the default CA locations when verifying S/MIME signed messages. Only applicable if S/MIME support is built using OpenSSL.

smtp-use-starttls
Causes *mailx* to issue a STARTTLS command to make an SMTP session SSL/TLS encrypted. Not all servers support this command; because of common implementation defects, it cannot be automatically determined whether a server supports it or not.

ssl-no-default-ca
Do not load the default CA locations to verify SSL/TLS server certificates. Only applicable if SSL/TLS support is built using OpenSSL.

ssl-v2-allow
Accept SSLv2 connections. These are normally not allowed because this protocol version is insecure.
WARNING: on modern systems SSLv2 as well as SSLv3 are unavailable!

stealthmua
Inhibits the generation of the *'Message-Id:'* and *'User-Agent:'* header fields that include obvious references to *mailx*. There are two pitfalls associated with this: First, the message id of outgoing messages is not known anymore. Second, an expert may still use the remaining information in the header to track down the originating mail user agent.

verbose
Setting the option verbose is the same as using the −v flag on the command line. When *mailx* runs in verbose mode, details of the actual message delivery and protocol conversations for IMAP, POP3, and SMTP, as well as of other internal processes, are displayed on the user's terminal, This is sometimes useful to debug problems. *Mailx* prints all data that is sent to remote servers in clear texts, including passwords, so care should be taken that no unauthorized option can view the screen if this option is enabled.

writebackedited
If this variable is set, messages modified using the *edit* or *visual* commands are written back to the current folder when it is quit. This is only possible for writable folders in *mbox* format. Setting this variable also disables MIME decoding and decryption for the editing commands.

String Options

The string options include the following:

attrlist A sequence of characters to print in the 'attribute' column of a header summary, each for one type of messages in the following order: new, unread but old, new but read, read and old, saved, preserved, mboxed, flagged, answered, draft, killed, start of a collapsed thread, collapsed, classified as junk. The default is 'NUROSPMFATK+−J', or 'NU *HMFATK+−J' if *bsdflags* or the *SYSV3* environment variable are set.

autobcc
Specifies a list of recipients to which a blind carbon copy of each outgoing message will be sent automatically.

autocc Specifies a list of recipients to which a carbon copy of each outgoing message will be sent automatically.

autosort
Causes sorted mode (see the *sort* command) to be entered automatically with the value of this option as sorting method when a folder is opened.

cmd The default value for the *pipe* command.

crt The valued option crt is used as a threshold to determine how long a message must be before PAGER is used to read it. If crt is set without a value, then the height of the terminal screen stored in the system is used to compute the threshold (see *stty*(1)).

DEAD The name of the file to use for saving aborted messages. This defaults to 'dead.letter' in the user's home directory.

EDITOR
Pathname of the text editor to use in the edit command and ˜e escape. If not defined, then a default editor is used.

encoding
The default MIME encoding to use in outgoing text messages and message parts. Valid values are *8bit* or *quoted-printable*. The default is *8bit*. In case the mail transfer system is not ESMTP compliant, *quoted-printable* should be used instead. If there is no need to encode a message, *7bit* transfer mode is used, without regard to the value of this variable. Binary data is always encoded in *base64* mode.

escape If defined, the first character of this option gives the character to use in the place of ˜ to denote escapes.

folder The name of the directory to use for storing folders of messages. All folder names that begin with '+' refer to files below that directory. If the directory name begins with a '/', *mailx* considers it to be an absolute pathname; otherwise, the folder directory is found relative to the user's home directory.

The directory name may also refer to an IMAP account; any names that begin with '+' then refer to IMAP mailboxes on that account. An IMAP folder is normally given in the form

```
imaps://mylogin@imap.myisp.example
```

In this case, the '+' and '@' prefixes for folder names have the same effect (see the *folder* command).

Some IMAP servers do not accept the creation of mailboxes in the hierarchy base; they require that they are created as subfolders of 'INBOX'. With such servers, a folder name of the form

```
imaps://mylogin@imap.myisp.example/INBOX.
```

should be used (the last character is the server's hierarchy delimiter). Folder names prefixed by '+' will then refer to folders below 'INBOX', while folder names prefixed by '@' refer to folders below the hierarchy base. See the *imap namespace* command for a method to detect the appropriate prefix and delimiter.

folder-hook
When a folder is opened and this variable is set, the macro corresponding to the value of this variable is executed. The macro is also invoked when new mail arrives, but message lists for commands executed from the macro only include newly arrived messages then.

folder-hook-*fullname*
When a folder named *fullname* is opened, the macro corresponding to the value of this variable is executed. Unlike other folder specifications, the fully expanded name of a folder, without metacharacters, is used to avoid ambiguities. The macro specified with *folder-hook* is not executed if this variable is effective for a folder (unless it is explicitly invoked within the called macro).

from The address (or a list of addresses) to put into the *'From:'* field of the message header. If replying to a message, these addresses are handled as if they were in the alternates list. If the machine's hostname is not valid at the Internet (for example at a dialup machine), either this variable or *hostname* have to be set to get correct Message-ID header fields. If *from* contains more than one address, the *sender* variable must also be set.

fwdheading
The string to print before the text of a message with the *forward* command (unless the *forward-as-attachment* variable is set). Defaults to "-------- Original Message --------" if unset. If it is set to the empty string, no heading is printed.

headline
A format string to use for the header summary, similar to *printf* formats. A '%' character introduces a format specifier. It may be followed by a number indicating the field width. If the field is a number, the width may be negative, which indicates that it is to be left-aligned. Valid format specifiers are:

%a	Message attributes.
%c	The score of the message.
%d	The date when the message was received.
%e	The indenting level in threaded mode.
%f	The address of the message sender.
%i	The message thread structure.
%l	The number of lines of the message.
%m	Message number.
%o	The number of octets (bytes) in the message.
%s	Message subject (if any).
%S	Message subject (if any) in double quotes.
%t	The position in threaded/sorted order.
%>	A '>' for the current message, otherwise ' '.
%<	A '<' for the current message, otherwise ' '.
%%	A '%' character.

The default is '%>%a%m %18f %16d %4l/%−5o %i%s', or '%>%a%m %20f %16d %3l/%−5o %i%S' if *bsdcompat* is set.

hostname
Use this string as hostname when expanding local addresses instead of the value obtained from *uname*(2) and *getaddrinfo*(3).

imap-auth
Sets the IMAP authentication method. Valid values are 'login' for the usual password-based authentication (the default), 'cram-md5', which is a password-based authentication that does not send the password over the network in clear text, and 'gssapi' for GSSAPI-based authentication.

imap-auth-*user*@*host*
Sets the IMAP authentication method for a specific account.

imap-cache
Enables caching of IMAP mailboxes. The value of this variable must point to a directory that is either existent or can be created by *mailx*. All contents of the cache can be deleted by *mailx* at any time; it is not safe to make assumptions about them.

imap-keepalive
IMAP servers may close the connection after a period of inactivity; the standard requires this to be at least 30 minutes, but practical experience may vary. Setting this variable to a numeric *value* greater than 0 causes a NOOP command to be sent each *value* seconds if no other operation is performed.

imap-list-depth
When retrieving the list of folders on an IMAP server, the *folders* command stops after it has reached a certain depth to avoid possible infinite loops. The value of this variable sets the maximum depth allowed. The default is 2. If the folder separator on the current IMAP server is a slash '/', this variable has no effect, and the *folders* command does not descend to subfolders.

indentprefix
String used by the '*~m*' and '*~M*' tilde escapes and by the *quote* option for indenting messages, in place of the normal tab character (^I). Be sure to quote the value if it contains spaces or tabs.

junkdb
The location of the junk mail database. The string is treated like a folder name, as described for the *folder* command.

The files in the junk mail database are normally stored in *compress*(1) format for saving space. If processing time is considered more important, *uncompress*(1) can be used to store them in plain form. *Mailx* will then work using the uncompressed files.

LISTER
Pathname of the directory lister to use in the *folders* command when operating on local mailboxes. Default is /bin/ls.

MAIL Is used as the user's mailbox, if set. Otherwise, a system-dependent default is used. Can be a *protocol***://** string (see the *folder* command for more information).

MAILX_HEAD
A string to put at the beginning of each new message. The escape sequences '**\t**' (tabulator) and '**\n**' (newline) are understood.

MAILX_TAIL
A string to put at the end of each new message. The escape sequences '**\t**' (tabulator) and '**\n**' (newline) are understood.

maximum-unencoded-line-length
Messages that contain lines longer than the value of this variable are encoded in quoted-printable even if they contain only ASCII characters. The maximum effective value is 950. If set to 0, all ASCII text messages are encoded in quoted-printable. S/MIME signed messages are always encoded in quoted-printable regardless of the value of this variable.

MBOX The name of the mbox file. It can be the name of a folder. The default is '*mbox*' in the user's home directory.

NAIL_EXTRA_RC
The name of an optional startup file to be read after ~/.mailrc. This variable is ignored if it is imported from the environment; it has an effect only if it is set in /etc/mail.rc or ~/.mailrc to allow bypassing the configuration with e. g. 'MAILRC=/dev/null'. Use this file for commands that are not understood by other mailx implementations.

newfolders
If this variable has the value **maildir**, newly created local folders will be in *maildir* format.

nss-config-dir
A directory that contains the files cert*N*.db to retrieve certificates, key*N*.db to retrieve private keys, and secmod.db, where *N* is a digit. These are usually taken from Mozilla installations, so an appropriate value might be '~/.mozilla/firefox/default.clm'. *Mailx* opens these files read-only and does not modify them. However, if the files are modified by Mozilla while *mailx* is running, it will print a 'Bad database' message. It may be necessary to create copies of these files that are exclusively used by *mailx* then. Only applicable if S/MIME and SSL/TLS support is built using Network Security Services (NSS).

ORGANIZATION
The value to put into the *'Organization:'* field of the message header.

PAGER
Pathname of the program to use in the more command or when crt variable is set. The default paginator *pg*(1) or, in BSD compatibility mode, *more*(1) is used if this option is not defined.

password-*user@host*
Set the password for *user* when connecting to *host*. If no such variable is defined for a host, the user will be asked for a password on standard input. Specifying passwords in a startup file is generally a security risk, the file should be readable by the invoking user only.

pipe-*content/subcontent*
When a MIME message part of *content/subcontent* type is displayed or it is replied to, its text is filtered through the value of this variable interpreted as a shell command. Special care must be taken when using such commands as mail viruses may be distributed by this method; if messages of type *application/x-sh* were filtered through the shell, for example, a message sender could easily execute arbitrary code on the system *mailx* is running on.

pop3-keepalive
POP3 servers may close the connection after a period of inactivity; the standard requires this to be at least 10 minutes, but practical experience may vary. Setting this variable to a numeric *value* greater than 0 causes a NOOP command to be sent each *value* seconds if no other operation is performed.

prompt
The string printed when a command is accepted. Defaults to '**?** ', or to '**&** ' if the *bsdcompat* variable is set.

quote If set, *mailx* starts a replying message with the original message prefixed by the value of the variable *indentprefix*. Normally, a heading consisting of 'Fromheaderfield wrote:' is printed before the quotation. If the string *noheading* is assigned to the *quote* variable, this heading is omitted. If the string *headers* is assigned, the headers selected by the ignore/retain commands are printed above the message body, thus *quote* acts like an automatic ~m command then. If the string *allheaders* is assigned, all headers are printed above the message body, and all MIME parts are included, thus *quote* acts like an automatic ~M command then.

record If defined, gives the pathname of the folder used to record all outgoing mail. If not defined, then outgoing mail is not so saved. When saving to this folder fails, the message is not sent but saved to the 'dead.letter' file instead.

replyto A list of addresses to put into the *'Reply-To:'* field of the message header. If replying to a message, such addresses are handled as if they were in the alternates list.

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

sendcharsets
A comma-separated list of character set names that can be used in Internet mail. When a message that contains characters not representable in US-ASCII is prepared for sending, *mailx* tries to convert its text to each of the given character sets in order and uses the first appropriate one. The default is 'utf-8'.

Character sets assigned to this variable should be ordered in ascending complexity. That is, the list should start with e. g. 'iso-8859-1' for compatibility with older mail clients, might contain some other language-specific character sets, and should end with 'utf-8' to handle messages that combine texts in multiple languages.

sender An address that is put into the 'Sender:' field of outgoing messages. This field needs not normally be present. It is, however, required if the 'From:' field contains more than one address. It can also be used to indicate that a message was sent on behalf of somebody other; in this case, 'From:' should contain the address of the person that took responsibility for the message, and 'Sender:' should contain the address of the person that actually sent the message. The *sender* address is handled as if it were in the *alternates* list.

sendmail
To use an alternate mail delivery system, set this option to the full pathname of the program to use. This should be used with care.

SHELL
Pathname of the shell to use in the ! command and the ~! escape. A default shell is used if this option is not defined.

Sign A string for use with the *~A* command.

sign A string for use with the *~a* command.

signature
Must correspond to the name of a readable file if set. The file's content is then appended to each singlepart message and to the first part of each multipart message. Be warned that there is no possibility to edit the signature for an individual message.

smime-ca-dir
Specifies a directory with CA certificates for verification of S/MIME signed messages. The format is the same as described in *SSL_CTX_load_verify_locations*(3). Only applicable if S/MIME support is built using OpenSSL.

smime-ca-file
Specifies a file with CA certificates for verification of S/MIME signed messages. The format is the same as described in *SSL_CTX_load_verify_locations*(3). Only applicable if S/MIME support is built using OpenSSL.

smime-cipher-*user@host*
Specifies a cipher to use when generating S/MIME encrypted messages for *user@host*. Valid ciphers are **rc2-40** (RC2 with 40 bits), **rc2-64** (RC2 with 64 bits), **des** (DES, 56 bits) and **des-ede3** (3DES, 112/168 bits). The default is 3DES. It is not recommended to use the other ciphers unless a recipient's client is actually unable to handle 3DES since they are comparatively weak; but even so, the recipient should upgrade his software in preference.

smime-crl-file
Specifies a file that contains a CRL in PEM format to use when verifying S/MIME messages. Only applicable if S/MIME support is built using OpenSSL.

smime-crl-dir
Specifies a directory that contains files with CRLs in PEM format to use when verifying S/MIME messages. Only applicable if S/MIME support is built using OpenSSL.

smime-encrypt-*user@host*
If this variable is set, messages to *user@host* are encrypted before sending. If S/MIME support is built using OpenSSL, the value of the variable must be set to the name of a file that contains a certificate in PEM format. If S/MIME support is built using NSS, the value of this variable is ignored, but if multiple certificates for *user@host* are available, the *smime-nickname-user@host* variable should be set. Otherwise a certificate for the recipient is automatically retrieved from the certificate database, if possible.

If a message is sent to multiple recipients, each of them for whom a corresponding variable is set will receive an individually encrypted message; other recipients will continue to receive the message in plain text unless the *smime-force-encryption* variable is set. It is recommended to sign encrypted messages, i. e. to also set the *smime-sign* variable.

smime-nickname-*user@host*
Specifies the nickname of a certificate to be used when encrypting messages for *user@host* . Only applicable if S/MIME support is built using NSS.

smime-sign-cert
Points to a file in PEM format that contains the user's private key as well as his certificate. Both are used with S/MIME for signing and decrypting messages. Only applicable if S/MIME support is built using OpenSSL.

smime-sign-cert-*user@host*
Overrides *smime-sign-cert* for the specific addresses. When signing messages and the value of the *from* variable is set to *user@host*, the specific file is used. When decrypting messages, their recipient fields (To: and Cc:) are searched for addresses for which such a variable is set. *Mailx* always uses the first address that matches, so if the same message is sent to more than one of the user's addresses using different encryption keys, decryption might fail. Only applicable if S/MIME support is built using OpenSSL.

smime-sign-nickname
Specifies that the named certificate be used for signing mail. If this variable is not set, but a single certificate matching the current *from* address is found in the database, that one is used automatically. Only applicable if S/MIME support is built using NSS.

smime-sign-nickname-*user@host*
Overrides *smime-sign-nickname* for a specific address. Only applicable if S/MIME support is built using NSS.

smtp Normally, *mailx* invokes *sendmail*(8) directly to transfer messages. If the *smtp* variable is set, a SMTP connection to the server specified by the value of this variable is used instead. If the SMTP server does not use the standard port, a value of *server:port* can be given, with *port* as a name or as a number.

There are two possible methods to get SSL/TLS encrypted SMTP sessions: First, the STARTTLS command can be used to encrypt a session after it has been initiated, but before any user-related data has been sent; see *smtp-use-starttls* above. Second, some servers accept sessions that are encrypted from their beginning on. This mode is configured by assigning **smtps://***server*[**:***port*] to the *smtp* variable.

The SMTP transfer is executed in a child process; unless either the *sendwait* or the *verbose* variable is set, this process runs asynchronously. If it receives a TERM signal, it will abort and save the message to the 'dead.letter' file.

smtp-auth
Sets the SMTP authentication method. If set to 'login', or if unset and smtp-auth-user is set, AUTH LOGIN is used. If set to 'cram-md5', AUTH CRAM-MD5 is used; if set to 'plain', AUTH PLAIN is used. Otherwise, no SMTP authentication is performed.

smtp-auth-*user@host*
Overrides *smtp-auth* for specific values of sender addresses, depending on the *from* variable.

smtp-auth-password
Sets the global password for SMTP AUTH. Both user and password have to be given for AUTH LOGIN and AUTH CRAM-MD5.

smtp-auth-password-*user@host*
Overrides *smtp-auth-password* for specific values of sender addresses, depending on the *from* variable.

smtp-auth-user
Sets the global user name for SMTP AUTH. Both user and password have to be given for AUTH LOGIN and AUTH CRAM-MD5.

If this variable is set but neither *smtp-auth-password* or a matching *smtp-auth-password-user@host* can be found, *mailx* will as for a password on the user's terminal.

smtp-auth-user-*user@host*
Overrides *smtp-auth-user* for specific values of sender addresses, depending on the *from* variable.

ssl-ca-dir
Specifies a directory with CA certificates for verification of SSL/TLS server certificates. See *SSL_CTX_load_verify_locations*(3) for more information. Only applicable if SSL/TLS support is built using OpenSSL.

ssl-ca-file
Specifies a file with CA certificates for verification of SSL/TLS server certificates. See *SSL_CTX_load_verify_locations*(3) for more information. Only applicable if SSL/TLS support is built using OpenSSL.

ssl-cert Sets the file name for a SSL/TLS client certificate required by some servers. Only applicable if SSL/TLS support is built using OpenSSL.

ssl-cert-*user@host*
Sets an account-specific file name for a SSL/TLS client certificate required by some servers. Overrides *ssl-cert* for the specified account. Only applicable if SSL/TLS support is built using OpenSSL.

ssl-cipher-list
Specifies a list of ciphers for SSL/TLS connections. See ciphers(1) for more information. Only applicable if SSL/TLS support is built using OpenSSL.

ssl-crl-file
Specifies a file that contains a CRL in PEM format to use when verifying SSL/TLS server certificates. Only applicable if SSL/TLS support is built using OpenSSL.

ssl-crl-dir
Specifies a directory that contains files with CRLs in PEM format to use when verifying SSL/TLS server certificates. Only applicable if SSL/TLS support is built using OpenSSL.

ssl-key Sets the file name for the private key of a SSL/TLS client certificate. If unset, the name of the certificate file is used. The file is expected to be in PEM format. Only applicable if SSL/TLS support is built using OpenSSL.

ssl-key-*user@host*
Sets an account-specific file name for the private key of a SSL/TLS client certificate. Overrides *ssl-key* for the specified account. Only applicable if SSL/TLS support is built using OpenSSL.

ssl-method
Selects a SSL/TLS protocol version; valid values are 'ssl2', 'ssl3', and 'tls1'. If unset, the method is selected automatically, if possible.
WARNING: Do not use this option. 'ssl2', 'ssl3' are no longer available and 'tls1' forces use of TLS 1.0

ssl-method-*user@host*
Overrides *ssl-method* for a specific account.

ssl-rand-egd
Gives the pathname to an entropy daemon socket, see *RAND_egd*(3).
WARNING: On Linux this API is unavailable.

ssl-rand-file
Gives the pathname to a file with entropy data, see *RAND_load_file*(3). If the file is a regular file writable by the invoking user, new data is written to it after it has been loaded. Only applicable if SSL/TLS support is built using OpenSSL.
WARNING: On linux the CSPRNG is seeded automatically and this option has no effect.

ssl-verify
Sets the action to be performed if an error occurs during SSL/TLS server certificate validation. Valid values are 'strict' (fail and close connection immediately), 'ask' (ask whether to continue on standard input), 'warn' (print a warning and continue), 'ignore' (do not perform validation). The default is 'ask'.

ssl-verify-*user@host*
Overrides *ssl-verify* for a specific account.

toplines
If defined, gives the number of lines of a message to be printed out with the top command; normally, the first five lines are printed.

ttycharset
The character set of the terminal *mailx* operates on. There is normally no need to set this variable since *mailx* can determine this automatically by looking at the LC_CTYPE locale setting; if this succeeds, the value is assigned at startup and will be displayed by the *set* command. Note that this is not necessarily a character set name that can be used in Internet messages.

VISUAL
Pathname of the text editor to use in the visual command and ˜v escape.

ENVIRONMENT VARIABLES

Besides the variables described above, *mailx* uses the following environment strings:

HOME
The user's home directory.

LANG, **LC_ALL**, **LC_COLLATE**, **LC_CTYPE**, **LC_MESSAGES**
See *locale*(7).

MAILRC
Is used as startup file instead of ˜/.mailrc if set. When *mailx* scripts are invoked on behalf of other users, this variable should be set to '/dev/null' to avoid side-effects from reading their configuration files.

NAILRC
If this variable is set and *MAILRC* is not set, it is read as startup file.

SYSV3 Changes the letters printed in the first column of a header summary.

TMPDIR
Used as directory for temporary files instead of /tmp, if set.

FILES

˜/.mailrc
File giving initial commands.

/etc/mail.rc
System wide initialization file.

˜/.mime.types
Personal MIME types.

/etc/mime.types
System wide MIME types.

EXAMPLES

Getting started

The *mailx* 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 email address is, say, <bill@host.example>, use the shell command:

$ **mailx** *bill@host.example*

then type your message. *Mailx* will prompt you for a message *subject* first; after that, lines typed by you form the body of the message. When you reach the end of the message, type an EOT (control−d) at the beginning of a line, which will cause *mailx* to echo ‘EOT’ and return you to the shell.

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 *mailx* to print ‘(Interrupt -- one more to kill letter)’. Typing a second RUBOUT causes *mailx* 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.

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

```
$ mailx sam@workstation.example bob@server.example
Subject: Fees
Tuition fees are due next Friday. Don't forget!
<Control−d>
EOT
$
```

will send the reminder to <*sam@workstation.example*>. and <*bob@server.example*>.

To read your mail, simply type

```
$ mailx
```

Mailx 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. *Mailx* 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. *Mailx* 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:

```
O 1 drfoo@myhost.example Wed Sep  1 19:52  18/631 "Fees"
O 2 sam@friends.example  Thu Sep  2 00:08  30/895
```

you could examine the first message by giving the command:

```
type 1
```

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

```
Message  1:
From drfoo@myhost.example Wed Sep  1 19:52:25 2004
Subject: Fees
Status: R

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

Many *mailx* 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 *mailx* program, the current message is initially the first (or the first recent) 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 *mailx* by simply typing a newline. As a special case, you can type a newline as your first command to *mailx* 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. This command, like *type*, takes a message number as an argument. *mailx* 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.

Note that *mailx* 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, like 'Cc:', the information found will also be used.

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@machine.example>, you would do:

mail *frank@machine.example*

To delete a message from the mail folder, you can use the *delete* command. In addition to not saving deleted messages, *mailx* will not let you type them, either. The effect is to make the message disappear altogether, along with its number.

Many features of *mailx* 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 *askcc* option informs *mailx* that each time you send a message, you want it to prompt you for a 'Cc:' header, to be included in the message. To set the *askcc* option, you would type

set askcc

Valued options are values which *mailx* uses to adapt to your tastes. For example, the *record* option tells *mailx* where to save messages sent by you, and is specified by

set record=Sent

for example. Note that no spaces are allowed in *set record=Sent* .

Mailx includes a simple facility for maintaining groups of messages together in folders. To use the folder facility, you must tell *mailx* 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 *mailx* 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 '/', *mailx* will assume that your folder directory is to be found starting from your home directory.

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.

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

folder +*classwork*

directs *mailx* 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.

Finally, the *help* command is available to print out a brief summary of the most important *mailx* commands.

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. *Mailx* 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. A list of the most important tilde escapes is available with '˜?'.

IMAP or POP3 client setup

First you need the following data from your ISP: the host name of the IMAP or POP3 server, user name and password for this server, and a notice whether the server uses SSL/TLS encryption. Assuming the host name is 'server.myisp.example' and your user name for that server is 'mylogin', you can refer to this account using the *folder* command or *−f* command line option with

imaps://*mylogin@server.myisp.example*

(This string is not necessarily the same as your Internet mail address.) You can replace 'imaps://' with 'imap://' if the server does not support SSL/TLS. (If SSL/TLS support is built using NSS, the *nss-config-dir* variable must be set before a connection can be initiated, see above). Use 'pop3s://' or 'pop3://' if the server does not offer IMAP. You should use IMAP if you can, though; first because it requires fewer network operations than POP3 to get the contents of the mailbox and is thus faster; and second because message attributes are maintained by the IMAP server, so you can easily distinguish new and old messages each time you connect. Even if the server does not accept IMAPS or POP3S connections, it is possible that it supports the STARTTLS method to make a session SSL/TLS encrypted after the initial connection has been performed, but before authentication begins. The only reliable method to see if this works is to try it; enter one of

set imap-use-starttls

set pop3-use-starttls

before you initiate the connection.

As you probably want messages to be deleted from this account after saving them, prefix it with '*%:*'. The *shortcut* command can be used to avoid typing that many characters every time you want to connect:

shortcut *myisp* **%:imaps://***mylogin@server.myisp.example*

You might want to put this string into a startup file. As the *shortcut* command is specific to this implementation of *mailx* and will confuse other implementations, it should not be used in *~/.mailrc*, instead, put

set NAIL_EXTRA_RC=*~/.nailrc*

in *~/.mailrc* and create a file *~/.nailrc* containing the *shortcut* command above. You can then access your remote mailbox by invoking 'mailx −f *myisp*' on the command line, or by executing 'fi *myisp*' within mailx.

If you want to use more than one IMAP mailbox on a server, or if you want to use the IMAP server for mail storage too, the *account* command (which is also *mailx*-specific) is more appropriate than the *shortcut* command. You can put the following in *~/.nailrc*:

account *myisp* **{**
  set folder=imaps://*mylogin@server.myisp.example*
  set record=+*Sent* **MBOX=+***mbox* **outfolder**
}

and can then access incoming mail for this account by invoking 'mailx −A *myisp*' on the command line, or by executing 'ac *myisp*' within mailx. After that, a command like 'copy *1 +otherfolder*' will refer to *otherfolder* on the IMAP server. In particular, 'fi &' will change to the *mbox* folder, and 'fi +Sent' will show your recorded sent mail, with both folders located on the IMAP server.

Mailx will ask you for a password string each time you connect to a remote account. If you can reasonably trust the security of your workstation, you can give this password in the startup file as

set password-*mylogin@server.myisp.example***="***SECRET***"**

You should change the permissions of this file to 0600, see *chmod*(1).

Mailx supports different authentication methods for both IMAP and POP3. If Kerberos is used at your location, you can try to activate GSSAPI-based authentication by

set imap-auth=gssapi

The advantage of this method is that *mailx* does not need to know your password at all, nor needs to send sensitive data over the network. Otherwise, the options

set imap-auth=cram-md5
set pop3-use-apop

for IMAP and POP3, respectively, offer authentication methods that avoid to send the password in clear text over the network, which is especially important if SSL/TLS cannot be used. If the server does not offer any of these authentication methods, conventional user/password based authentication must be used. It is sometimes helpful to set the *verbose* option when authentication problems occur. *Mailx* will display all data sent to the server in clear text on the screen with this option, including passwords. You should thus take care that no unauthorized person can look at your terminal when this option is set.

If you regularly use the same workstation to access IMAP accounts, you can greatly enhance performance by enabling local caching of IMAP messages. For any message that has been fully or partially fetched from the server, a local copy is made and is used when the message is accessed again, so most data is transferred over the network once only. To enable the IMAP cache, select a local directory name and put

set imap-cache=~*/localdirectory*

in the startup file. All files within that directory can be overwritten or deleted by *mailx* at any time, so you should not use the directory to store other information.

Once the cache contains some messages, it is not strictly necessary anymore to open a connection to the IMAP server to access them. When *mailx* is invoked with the *−D* option, or when the *disconnected* variable is set, only cached data is used for any folder you open. Messages that have not yet been completely cached are not available then, but all other messages can be handled as usual. Changes made to IMAP mailboxes in *disconnected* mode are committed to the IMAP server next time it is used in *online* mode. Synchronizing the local status with the status on the server is thus partially within your responsibility; if you forget to initiate a connection to the server again before you leave your location, changes made on one workstation are not available on others. Also if you alter IMAP mailboxes from a workstation while uncommitted changes are still pending on another, the latter data may become invalid. The same might also happen because of internal server status changes. You should thus carefully evaluate this feature in your environment before you rely on it.

Many servers will close the connection after a short period of inactivity. Use one of

set pop3-keepalive=*30*
set imap-keepalive=*240*

to send a keepalive message each 30 seconds for POP3, or each 4 minutes for IMAP.

If you encounter problems connecting to a SSL/TLS server, try the *ssl-rand-egd* and *ssl-rand-file* variables (see the OpenSSL FAQ for more information) or specify the protocol version with *ssl-method*. Contact your ISP if you need a client certificate or if verification of the server certificate fails. If the failed certificate is indeed valid, fetch its CA certificate by executing the shell command

$ **openssl s_client </dev/null −showcerts −connect **
*server.myisp.example***:***imaps* **2>&1 | tee** *log*

(see *s_client*(1)) and put it into the file specified with *ssl-ca-file*. The data you need is located at the end of the certificate chain within (and including) the 'BEGIN CERTIFICATE' and 'END CERTIFICATE' lines. (Note that it is possible to fetch a forged certificate by this method. You can only completely rely on the authenticity of the CA certificate if you fetch it in a way that is trusted by other means, such as by personally receiving the certificate on storage media.)

Creating a score file or message filter

The scoring commands are best separated from other configuration for clarity, and are mostly *mailx* specific. It is thus recommended to put them in a separate file that is sourced from your NAIL_EXTRA_RC as follows:

source ~*/.scores*

The *.scores* file could then look as follows:

define *list* {
score (subject "important discussion") +10
score (subject "annoying discussion") −10
score (from "nicefellow@goodnet") +15
score (from "badguy@poornet") −5
move (header x-spam-flag "+++++") +*junk*

```
}
set folder-hook-imap://user@host/public.list=list
```

In this scheme, you would see any mail from 'nicefellow@goodnet', even if the surrounding discussion is annoying; but you normally would not see mail from 'badguy@poornet', unless he participates in the important discussion. Messages that are marked with five or more plus characters in their 'X-Spam-Flag' field (inserted by some server-side filtering software) are moved to the folder 'junk' in the *folder* directory.

Be aware that all criteria in ( ) lead to substring matches, so you would also score messages from e. g. 'notsobadguy@poornetmakers' negative here. It is possible to select addresses exactly using *"address"* message specifications, but these cannot be executed remotely and will thus cause all headers to be downloaded from IMAP servers while looking for matches.

When searching messages on an IMAP server, best performance is usually achieved by sending as many criteria as possible in one large ( ) specification, because each single such specification will result in a separate network operation.

Activating the Bayesian filter

The Bayesian junk mail filter works by examining the words contained in messages. You decide yourself what a good and what a bad message is. Thus the resulting filter is your very personal one; once it is correctly set up, it will filter only messages similar to those previously specified by you.

To use the Bayesian filter, a location for the junk mail database must be defined first:

```
set junkdb=~/.junkdb
```

The junk mail database does not contain actual words extracted from messages, but hashed representations of them. A foreign person who can read the database could only examine the frequency of previously known words in your mail.

If you have sufficient disk space (several 10 MB) available, it is recommended that you set the *chained-junk-tokens* option. The filter will then also consider two-word tokens, improving its accuracy.

A set of good messages and junk messages must now be available; it is also possible to use the incoming new messages for this purpose, although it will of course take some time until the filter becomes useful then. Do not underestimate the amount of statistical data needed; some hundred messages are typically necessary to get satisfactory results, and many thousand messages for best operation. You have to pass the good messages to the *good* command, and the junk messages to the *junk* command. If you ever accidentally mark a good message as junk or vice-versa, call the *ungood* or *unjunk* command to correct this.

Once a reasonable amount of statistics has been collected, new messages can be classified automatically. The *classify* command marks all messages that the filter considers to be junk, but it does not perform any action on them by default. It is recommended that you move these messages into a separate folder just for the case that false positives occur, or to pass them to the *junk* command later again to further improve the junk mail database. To automatically move incoming junk messages every time the inbox is opened, put lines like the following into your *.scores* file (or whatever name you gave to the file in the last example):

```
define junkfilter {
   classify (smaller 20000) :n
   move :j +junk
}
set folder-hook-imap://user@host/INBOX=junkfilter
```

If you set the *verbose* option before running the *classify* command, *mailx* prints the words it uses for calculating the junk status along with their statistical probabilities. This can help you to find out why some messages are not classified as you would like them to be. To see the statistical probability of a given word, use the *probability* command.

If a junk message was not recognized as such, use the *junk* command to correct this. Also if you encounter a false positive (a good message that was wrongly classified as junk), pass it to the *good* command.

Since the *classify* command must examine the entire text of all new messages in the respective folder, this will also cause all of them to be downloaded from the IMAP server. You should thus restrict the size of messages for automatic filtering. If server-based filtering is also available, you might try if that works for you first.

Reading HTML mail

You need either the *w3m* or *lynx* utility or another command-line web browser that can write plain text to standard output.

set pipe-text/html="w3m -dump -T text/html"

or

set pipe-text/html="lynx -dump -force_html /dev/stdin"

will then cause HTML message parts to be converted into a more friendly form.

Viewing PDF attachments

Most PDF viewers do not accept input directly from a pipe. It is thus necessary to store the attachment in a temporary file, as with

set pipe-application/pdf="cat >/tmp/mailx$$.pdf; \
acroread /tmp/mailx$$.pdf; rm /tmp/mailx$$.pdf"

Note that security defects are discovered in PDF viewers from time to time. Automatical command execution like this can compromise your system security, in particular if you stay not always informed about such issues.

Signed and encrypted messages with S/MIME

S/MIME provides two central mechanisms: message signing and message encryption. A signed message contains some data in addition to the regular text. The data can be used to verify that the message was sent using a valid certificate, that the sender's address in the message header matches that in the certificate, and that the message text has not been altered. Signing a message does not change its regular text; it can be read regardless of whether the recipient's software is able to handle S/MIME. It is thus usually possible to sign all outgoing messages if so desired.—Encryption, in contrast, makes the message text invisible for all people except those who have access to the secret decryption key. To encrypt a message, the specific recipient's public encryption key must be known. It is thus not possible to send encrypted mail to people unless their key has been retrieved from either previous communication or public key directories. A message should always be signed before it is encrypted. Otherwise, it is still possible that the encrypted message text is altered.

A central concept to S/MIME is that of the certification authority (CA). A CA is a trusted institution that issues certificates. For each of these certificates, it can be verified that it really originates from the CA, provided that the CA's own certificate is previously known. A set of CA certificates is usually delivered with OpenSSL and installed on your system. If you trust the source of your OpenSSL software installation, this offers reasonable security for S/MIME on the Internet. In general, a certificate cannot be more secure than the method its CA certificate has been retrieved with, though. Thus if you download a CA certificate from the Internet, you can only trust the messages you verify using that certificate as much as you trust the download process.

The first thing you need for participating in S/MIME message exchange is your personal certificate, including a private key. The certificate contains public information, in particular your name and your email address, and the public key that is used by others to encrypt messages for you, and to verify signed messages they supposedly received from you. The certificate is included in each signed message you send. The private key must be kept secret. It is used to decrypt messages that were previously encrypted with your public key, and to sign messages.

For personal use, it is recommended that you get a S/MIME certificate from one of the major CAs on the Internet using your WWW browser. (Many CAs offer such certificates for free.) You will usually receive a combined certificate and private key in PKCS#12 format which *mailx* does not directly accept if S/MIME support is built using OpenSSL. To convert it to PEM format, use the following shell command:

$ **openssl pkcs12 −in** *cert.p12* **−out** *cert.pem* **−clcerts **
−nodes

If you omit the *−nodes* parameter, you can specifiy an additional *PEM pass phrase* for protecting the private key. *Mailx* will then ask you for that pass phrase each time it signs or decrypts a message. You can then use

set smime-sign-cert-*myname@myisp.example***=***cert.pem*

to make this private key and certificate known to *mailx*.

If S/MIME support is built using NSS, the PKCS#12 file must be installed using Mozilla (provided that *nss-config-dir* is set appropriately, see above), and no further action is necessary unless multiple user certificates for the same email address are installed. In this case, the *smime-sign-nickname* variable has to be set appropriately.

You can now sign outgoing messages. Just use

set smime-sign

to do so.

From each signed message you send, the recipient can fetch your certificate and use it to send encrypted mail back to you. Accordingly if somebody sends you a signed message, you can do the same. First use the *verify* command to check the validity of the certificate. After that, retrieve the certificate and tell *mailx* that it should use it for encryption:

certsave *filename*
set smime-encrypt-*user@host***=***filename*

If S/MIME support is built using NSS, the saved certificate must be installed using Mozilla. The value of the *smime-encrypt-user@host* is ignored then, but if multiple certificates for the recipient are available, the *smime-nickname-user@host* variable must be set.

You should carefully consider if you prefer to store encrypted messages in decrypted form. If you do, anybody who has access to your mail folders can read them, but if you do not, you might be unable to read them yourself later if you happen to lose your private key. The *decrypt* command saves messages in decrypted form, while the *save*, *copy*, and *move* commands leave them encrypted.

Note that neither S/MIME signing nor encryption applies to message subjects or other header fields. Thus they may not contain sensitive information for encrypted messages, and cannot be trusted even if the message content has been verified. When sending signed messages, it is recommended to repeat any important header information in the message text.

Using CRLs with S/MIME or SSL/TLS

Certification authorities (CAs) issue certificate revocation lists (CRLs) on a regular basis. These lists contain the serial numbers of certificates that have been declared invalid after they have been issued. Such usually happens because the private key for the certificate has been compromised, because the owner of the certificate has left the organization that is mentioned in the certificate, etc. To seriously use S/MIME or SSL/TLS verification, an up-to-date CRL is required for each trusted CA. There is otherwise no method to distinguish between valid and invalidated certificates. *Mailx* currently offers no mechanism to fetch CRLs, or to access them on the Internet, so you have to retrieve them by some external mechanism.

If S/MIME and SSL/TLS support are built using OpenSSL, *mailx* accepts CRLs in PEM format only;

CRLs in DER format must be converted, e. g. with the shell command

$ **openssl crl −inform DER −in** *crl.der* **−out** *crl.pem*

To tell *mailx* about the CRLs, a directory that contains all CRL files (and no other files) must be created. The *smime-crl-dir* or *ssl-crl-dir* variables, respectively, must then be set to point to that directory. After that, *mailx* requires a CRL to be present for each CA that is used to verify a certificate.

If S/MIME and SSL/TLS support are built using NSS, CRLs can be imported in Mozilla applications (provided that *nss-config-dir* is set appropriately).

Sending mail from scripts

If you want to send mail from scripts, you must be aware that *mailx* reads the user's configuration files by default. So unless your script is only intended for your own personal use (as e.g. a cron job), you need to circumvent this by invoking *mailx* like

MAILRC=/dev/null mailx −n

You then need to create a configuration for *mailx* for your script. This can be done by either pointing the *MAILRC* variable to a custom configuration file, or by passing the configuration in environment variables. Since many of the configuration options are not valid shell variables, the *env* command is useful in this situation. An invocation could thus look like

env MAILRC=/dev/null from=*scriptreply@domain* smtp=*host* \
smtp-auth-user=*login* smtp-auth-password=*secret* \
smtp-auth=*login* **mailx −n** −s "*subject*" \
−a *attachment_file recipient@domain* <*content_file*

SEE ALSO

fmt(1), newaliases(1), openssl(1), pg(1), more(1), vacation(1), ssl(3), aliases(5), locale(7), mailaddr(7), sendmail(8)

NOTES

If used in systemd unit files not using **Type=forking** it is required to set the option *sendwait* on the command line of **mailx**(1) with *−S sendwait=yes* as otherwise it might happen that the **systemd**(1) service manager will terminate the asynchronously started SMTP transfer as described at the option *smtp.*

Variables in the environment passed to *mailx* cannot be unset.

The character set conversion relies on the *iconv*(3) function. Its functionality differs widely between the various system environments *mailx* runs on. If the message 'Cannot convert from *a* to *b*' appears, either some characters within the message header or text are not appropriate for the currently selected terminal character set, or the needed conversion is not supported by the system. In the first case, it is necessary to set an appropriate LC_CTYPE locale (e. g. *en_US*) or the *ttycharset* variable. In the second case, the *sendcharsets* and *ttycharset* variables must be set to the same value to inhibit character set conversion. If *iconv()* is not available at all, the value assigned to *sendcharsets* must match the character set that is used on the terminal.

Mailx expects input text to be in Unix format, with lines separated by *newline* (^J, \n) characters only. Non-Unix text files that use *carriage return* (^M, \r) characters in addition will be treated as binary data; to send such files as text, strip these characters e. g. by

```
tr −d '\015' <input | mailx . . .
```

or fix the tools that generate them.

Limitations with IMAP mailboxes are: It is not possible to edit messages, but it is possible to append them. Thus to edit a message, create a local copy of it, edit it, append it, and delete the original. The line count for the header display is only appropriate if the entire message has been downloaded from the server. The marking of messages as 'new' is performed by the IMAP server; use of the *exit* command instead of *quit* will not cause it to be reset, and if the *autoinc/newmail* variables are unset, messages that arrived during a session will not be in state 'new' anymore when the folder is opened again. Also if commands queued in disconnected mode are committed, the IMAP server will delete the 'new' flag for all messages in the changed folder, and new messages will appear as unread when it is selected for viewing later. The 'flagged', 'answered', and 'draft' attributes are usually permanent, but some IMAP servers are known to drop them without notification. Message numbers may change with IMAP every time before the prompt is printed if *mailx* is notified by the server that messages have been deleted by some other client or process. In this case, 'Expunged *n* messages' is printed, and message numbers may have changed.

Limitations with POP3 mailboxes are: It is not possible to edit messages, they can only be copied and deleted. The line count for the header display is only appropriate if the entire message has been downloaded from the server. The status field of a message is maintained by the server between connections; some servers do not update it at all, and with a server that does, the 'exit' command will not cause the message status to be reset. The 'newmail' command and the 'newmail' variable have no effect. It is not possible to rename or to remove POP3 mailboxes.

If a RUBOUT (interrupt) is typed while an IMAP or POP3 operation is in progress, *mailx* will wait until the operation can be safely aborted, and will then return to the command loop and print the prompt again. When a second *RUBOUT* is typed while *mailx* is waiting for the operation to complete, the operation itself will be canceled. In this case, data that has not been fetched yet will have to be fetched before the next command can be performed. If the canceled operation was using an SSL/TLS encrypted channel, an error in the SSL transport will very likely result, and the connection is no longer usable.

As *mailx* is a mail user agent, it provides only basic SMTP services. If it fails to contact its upstream SMTP server, it will not make further attempts to transfer the message at a later time, and it does not leave other information about this condition than an error message on the terminal and a 'dead.letter' file. This is usually not a problem if the SMTP server is located in the same local network as the computer on which *mailx* is run. However, care should be taken when using a remote server of an ISP; it might be better to set up a local SMTP server then which just acts as a proxy.

Mailx immediately contacts the SMTP server (or */usr/lib/sendmail*) even when operating in *disconnected* mode. It would not make much sense for *mailx* to defer outgoing mail since SMTP servers usually provide much more elaborated delay handling than *mailx* could perform as a client. Thus the recommended setup for sending mail in *disconnected* mode is to configure a local SMTP server such that it sends outgoing mail as soon as an external network connection is available again, i. e. to advise it to do that from a network startup script.

The junk mail filter follows the concepts developed by Paul Graham in his articles, ''A Plan for Spam'', August 2002, <http://www.paulgraham.com/spam.html>, and ''Better Bayesian Filtering'', January 2003, <http://www.paulgraham.com/better.html>. Chained tokens are due to a paper by Jonathan A. Zdziarski, ''Advanced Language Classification using Chained Tokens'', February 2004, <http://www.nuclearelephant.com/papers/chained.html>.

A *mail* command appeared in Version 1 AT&T Unix. Berkeley Mail was written in 1978 by Kurt Shoens. This man page is derived from from The Mail Reference Manual originally written by Kurt Shoens. *Heirloom Mailx* enhancements are maintained and documented by Gunnar Ritter.