mh-e

The Emacs Interface to MH

by Bill Wohler

Edition 1.1 for mh-e Version 5.0.1

April 1995

Preface

These chapters introduce another interface to MH that is accessible through the GNU Emacs editor, namely, *mh-e*. mh-e is easy to use. I don't assume that you know GNU Emacs or even MH at this point, since I didn't know either of them when I discovered mh-e. However, mh-e was the tip of the iceberg, and I discovered more and more niceties about GNU Emacs and MH. Now I'm fully hooked on both of them.

The mh-e package is distributed with GNU Emacs,[1] so you shouldn't have to do anything special to use it. But it's important to note a brief history of mh-e. Version 3 was prevalent through the Emacs 18 and early Emacs 19 years. Then Version 4 came out (Emacs 19.23), which introduced several new and changed commands. Finally, Version 5.0 was released, which fixed some bugs and incompatibilities. This is the version covered by this manual. Section 1.2 [Getting Started], page 4, will help you decide which version you have.

If you don't already use GNU Emacs but want to learn more, you can read an online tutorial by starting GNU Emacs and typing `C-h t` (`help-with-tutorial`). (This notation is described in Section 1.1 [Conventions], page 3.) If you want to take the plunge, consult the *GNU Emacs Manual*, from the Free Software Foundation.

If more information is needed, you can go to the Unix manual pages of the individual MH commands. When the name is not obvious, I'll guide you to a relevant MH manual page that describes the action more fully.

I hope you enjoy these chapters! If you have any comments, or suggestions for this document, please let me know.

Bill Wohler <*wohler@newt.com*>
8 February 1995

[1] Note that mh-e is supported with MH 6 and either GNU Emacs 18 or GNU Emacs 19. Reportedly, large parts of it work with MH 5 and also with Lucid/XEmacs and Epoch, but there are no guarantees. It is also distributed with Lucid/XEmacs, as well as with MH itself.

1 Tour Through mh-e

This chapter introduces some of the terms you'll need to know and then takes you on a tour of mh-e.[1] When you're done, you'll be able to send, read, and file mail, which is all that a lot of people ever do. But if you're the curious type, you'll read Chapter 2 [Using mh-e], page 11, to be able to use all the features of mh-e. If you're the adventurous type, you'll read Chapter 3 [Customizing mh-e], page 27, to make mh-e do what you want. I suggest you read this chapter first to get the big picture, and then you can read the other two as you wish.

1.1 GNU Emacs Terms and Conventions

If you're an experienced Emacs user, you can skip the following conventions and definition of terms and go directly to Section 1.2 [Getting Started], page 4, below. The conventions are as follows:

C-x	Hold down the `CTRL` (Control) key and press the `x` key.
M-x	Hold down the `META` or `ALT` key and press the `x` key. Since some keyboards don't have a `META` key, you can generate `M-x`, for example, by pressing `ESC` (Escape), *releasing it*,[2] and then pressing the `x` key.
RET	Press the `RETURN` or `ENTER` key. This is normally used to complete a command.
SPC	Press the space bar.
TAB	Press the `TAB` key.
DEL	Press the `DELETE` key. This may also be a Backspace key, depending on your keyboard or Emacs configuration.

A *prefix argument* allows you to pass an argument to any Emacs function. To pass an argument, type `C-u` before the Emacs command or keystroke. Numeric arguments can be passed as well. For example, to insert five f's, use `C-u 5 f`. There is a default of four when using `C-u`, and you can use multiple prefix arguments to provide arguments of powers of four. To continue our example, you could insert four f's with `C-u f`, 16 f's with `C-u C-u f`, 64 f's with `C-u C-u C-u f`, and so on. Numeric and valueless negative arguments can also be inserted with the `META` key. Examples include `M-5` to specify an argument of 5, or `M--` which specifies a negative argument with no particular value.

[1] The keys mentioned in these chapters refer to the default key bindings. If you've changed the bindings, refer to the command summaries at the beginning of each major section in Chapter 2 [Using mh-e], page 11, for a mapping between default key bindings and function names.

[2] This is emphasized because pressing ESC twice or holding it down a second too long so that it repeats gives you an error message.

NOTE

> The prefix *`C-u`* or *`M-`* is not necessary in mh-e's MH-Folder modes (see Section 1.4 [Reading Mail Tour], page 6). In these modes, simply enter the numerical argument before entering the command.

There are several other terms that are used in Emacs that you should know. The *point* is where the cursor currently is. You can save your current place in the file by setting a *mark*. This operation is useful in several ways. The mark can be later used when defining a *region*, which is the text between the point and mark. Many commands operate on regions, such as those for deleting text or filling paragraphs. A mark can be set with *`C-@`* (or *`C-SPC`*).

The *minibuffer* is the bottom line of the Emacs window, where all prompting and multiple-character input is directed. If you are prompted for information in the minibuffer, such as a filename, Emacs can help you complete your answer if you type `SPC` or `TAB`. A second `SPC` or `TAB` will list all possibilities at that point. The minibuffer is also where you enter Emacs function names after typing *`M-x`*. For example, in the first paragraph, I mentioned that you could obtain help with *`C-h t`* (`help-with-tutorial`). What this means is that you can get a tutorial by typing either *`C-h t`* or *`M-x help-with-tutorial`*. In the latter case, you are prompted for '`help-with-tutorial`' in the minibuffer after typing *`M-x`*.

In case of trouble: Emacs can be interrupted at any time with *`C-g`*. For example, if you've started a command that requests that you enter something in the minibuffer, but then you change your mind, type *`C-g`* and you'll be back where you started. If you want to exit Emacs entirely, use *`C-x C-c`*.

1.2 Getting Started

Because there are many old versions of mh-e out there, it is important to know which version you have. I'll be talking about Version 5 which is similar to Version 4 and vastly different from Version 3.

First, enter *`M-x load-library`* RET *`mh-e`* RET.[3] The message, '`Loading mh-e...done`', should be displayed in the minibuffer. If you get '`Cannot open load file: mh-e`', then your Emacs is very badly configured, or mh-e is missing. You may wish to have your system administrator install a new Emacs or at least the latest mh-e files.

Having loaded mh-e successfully, enter *`M-x mh-version`* RET. The version of mh-e should be displayed. Hopefully it says that you're running Version 5.0.1 which is the latest version as of this printing. If instead Emacs beeps and says '`[No match]`', then you're running an old version of mh-e.

If these tests reveal a non-existent or old version of mh-e, please consider obtaining a new version. You can have your system administrator upgrade the system-wide version, or you can install your own personal version. It's really quite easy; instructions for getting and installing mh-e are in Section A.4 [Getting mh-e], page 45. In the meantime, see Appendix C [Changes to mh-e], page 49, which compares the old and new names of commands, functions, variables, and buffers.

Also, older versions of mh-e assumed that you had already set up your MH environment. Newer versions set up a new MH environment for you by running `install-mh` and notifying you of this fact with the message in a temporary buffer:

[3] You wouldn't ordinarily do this.

```
I'm going to create the standard MH path for you.
```

Therefore, if you've never run MH before and you're using an old version of mh-e, you need to run `install-mh` from the shell before you continue the tour. If you don't, you'll be greeted with the error message: '`Can't find MH profile`'.

If, during the tour described in this chapter, you see a message like: '`Searching for program: no such file or directory, /usr/local/bin/mhpath`', it means that the MH programs and files are kept in a nonstandard directory. In this case, simply add the following to `~/.emacs` and restart `emacs`.

```
(setq mh-progs "/path/to/MH/binary/directory/")
(setq mh-lib "/path/to/MH/library/directory/")
```

The '`~`' notation used by `~/.emacs` above represents your home directory. This is used by the `bash` and `csh` shells. If your shell does not support this feature, you could use the environment variable '`$HOME`' (such as `$HOME/.emacs`) or the absolute path (as in `/home/wohler/.emacs`) instead.

At this point, you should see something like the screen in the figure in Section 1.4 [Reading Mail Tour], page 6. We're now ready to move on.

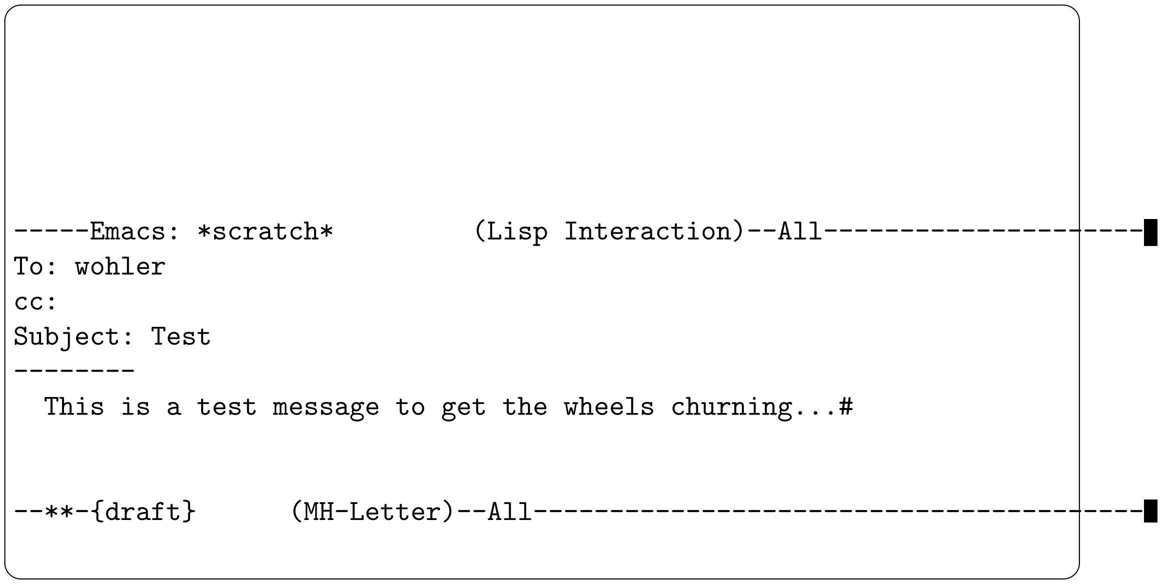
1.3 Sending Mail

Let's start our tour by sending ourselves a message which we can later read and process. Enter *`M-x mh-smail`* to invoke the mh-e program to send messages. You will be prompted in the minibuffer by '`To:`'. Enter your login name. The next prompt is '`cc:`'. Hit `RET` to indicate that no carbon copies are to be sent. At the '`Subject:`' prompt, enter *`Test`* or anything else that comes to mind.

Once you've specified the recipients and subject, your message appears in an Emacs buffer whose mode[4] is MH-Letter. Enter some text in the body of the message, using normal Emacs commands. You should now have something like this:[5]

[4] A *mode* changes Emacs to make it easier to edit a particular type of text.

[5] If you're running Emacs under the X Window System, then you would also see a menubar. I've left out the menubar in all of the example screens.



```
-----Emacs: *scratch*         (Lisp Interaction)--All-----------------------
To: wohler
cc:
Subject: Test
--------
  This is a test message to get the wheels churning...#


--**-{draft}      (MH-Letter)--All-----------------------------------------
```

mh-e message composition window

Note the line of dashes that separates the header and the body of the message. It is essential that these dashes (or a blank line) are present or the body of your message will be considered to be part of the header.

There are several commands specific to MH-Letter mode, but at this time we'll only use *C-c C-c* to send your message. Type *C-c C-c* now. That's all there is to it!

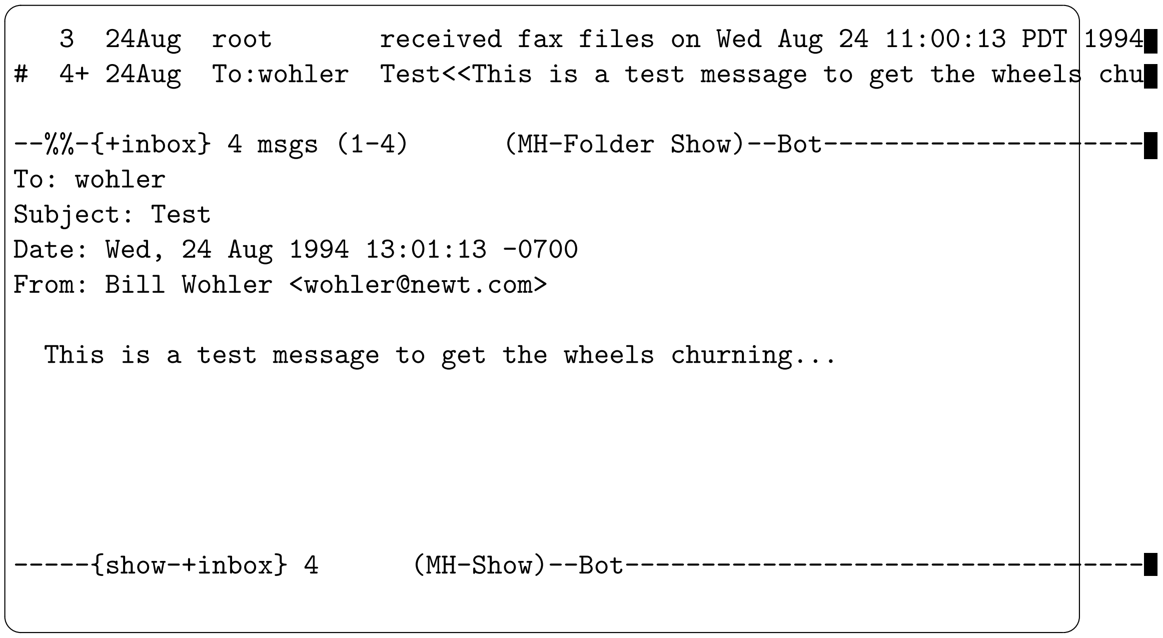
1.4 Receiving Mail

To read the mail you've just sent yourself, enter *M-x mh-rmail*. This incorporates the new mail and put the output from `inc` (called *scan lines* after the MH program `scan` which prints a one-line summary of each message) into a buffer called '`+inbox`' whose major mode is MH-Folder.

NOTE

> The *M-x mh-rmail* command will show you only new mail, not old mail. If you were to run this tour again, you would use *M-r* to pull all your messages into mh-e.

You should see the scan line for your message, and perhaps others. Use *n* or *p* to move the cursor to your test message and type `RET` to read your message. You should see something like:



```
   3  24Aug  root       received fax files on Wed Aug 24 11:00:13 PDT 1994
#  4+ 24Aug  To:wohler  Test<<This is a test message to get the wheels chu

--%%-{+inbox} 4 msgs (1-4)      (MH-Folder Show)--Bot---------------------
To: wohler
Subject: Test
Date: Wed, 24 Aug 1994 13:01:13 -0700
From: Bill Wohler <wohler@newt.com>

  This is a test message to get the wheels churning...





-----{show-+inbox} 4      (MH-Show)--Bot----------------------------------
```

After incorporating new messages

If you typed a long message, you can view subsequent pages with `SPC` and previous pages with `DEL`.

1.5 Processing Mail

The first thing we want to do is reply to the message that we sent ourselves. Ensure that the cursor is still on the same line as your test message and type *r*. You are prompted in the minibuffer with '`Reply to whom:`'. Here mh-e is asking whether you'd like to reply to the original sender only, to the sender and primary recipients, or to the sender and all recipients. If you simply hit `RET`, you'll reply only to the sender. Hit `RET` now.

You'll find yourself in an Emacs buffer similar to that when you were sending the original message, like this:

```
To: wohler
Subject: Re: Test
In-reply-to: Bill Wohler's message of Wed, 24 Aug 1994 13:01:13 -0700
             <199408242001.NAA00505@newt.com>
--------
#

--**-{draft}      (MH-Letter)--All---------------------------------------
To: wohler
Subject: Test
Date: Wed, 24 Aug 1994 13:01:13 -0700
From: Bill Wohler <wohler@newt.com>

  This is a test message to get the wheels churning...

-----{show-+inbox} 4      (MH-Show)--Bot---------------------------------
Composing a reply...done
```

Composition window during reply

By default, MH will not add you to the address list of your replies, so if you find that the '`To:`' header field is missing, don't worry. In this case, type *`C-c C-f C-t`* to create and go to the '`To:`' field, where you can type your login name again. You can move around with the arrow keys or with *`C-p`* (`previous-line`), *`C-n`* (`next-line`), *`C-b`* (`backward-char`), and *`C-f`* (`forward-char`) and can delete the previous character with `DEL`. When you're finished editing your message, send it with *`C-c C-c`* as before.

You'll often want to save messages that were sent to you in an organized fashion. This is done with *folders*. You can use folders to keep messages from your friends, or messages related to a particular topic. With your cursor in the MH-Folder buffer and positioned on the message you sent to yourself, type *`o`* to output (`refile` in MH parlance) that message to a folder. Enter *`test`* at the '`Destination:`' prompt and type *`y`* (or `SPC`) when mh-e asks to create the folder '`+test`'. Note that a '`^`' (caret) appears next to the message number, which means that the message has been marked for refiling but has not yet been refiled. We'll talk about how the refile is actually carried out in a moment.

Your previous reply is now waiting in the system mailbox. You incorporate this mail into your MH-Folder buffer named '`+inbox`' with the *`i`* command. Do this now. After the mail is incorporated, use *`n`* or *`p`* to move the cursor to the new message, and read it with `RET`. Let's delete this message by typing *`d`*. Note that a '`D`' appears next to the message number. This means that the message is marked for deletion but is not yet deleted. To perform the deletion (and the refile we did previously), use the *`x`* command.

If you want to send another message you can use *`m`* instead of *`M-x mh-smail`*. So go ahead, send some mail to your friends!

1.6 Leaving mh-e

You may now wish to exit `emacs` entirely. Use *C-x C-c* to exit `emacs`. If you exited without running *x* in the '`+inbox`' buffer, Emacs will offer to save it for you. Type *y* or `SPC` to save '`+inbox`' changes, which means to perform any refiles and deletes that you did there.

If you don't want to leave Emacs, you can type *q* to bury (hide) the mh-e folder or delete them entirely with *C-x k*. You can then later recall them with *C-x b* or *M-x mh-rmail*.

1.7 More About mh-e

These are the basic commands to get you going, but there are plenty more. If you think that mh-e is for you, read Chapter 2 [Using mh-e], page 11, and Chapter 3 [Customizing mh-e], page 27, to find out how you can:

- Print your messages. (Section 2.4.4 [Printing], page 21, and Section 3.4.4 [Customizing Printing], page 42.)
- Edit messages and include your signature. (Section 2.3 [Draft Editing], page 15, and Section 3.3 [Customizing Draft Editing], page 35.)
- Forward messages. (Section 2.2.2 [Forwarding], page 14, and Section 3.2.2 [Customizing Forwarding], page 34.)
- Read digests. (Section 2.1.1 [Viewing], page 12.)
- Edit bounced messages. (Section 2.2.4 [Old Drafts], page 14, and Section 3.2.4 [Customizing Old Drafts], page 35.)
- Send multimedia messages. (Section 2.3.2 [Editing MIME], page 17, and Section 3.3.2 [Customizing Editing MIME], page 37.)
- Process mail that was sent with `shar` or `uuencode`. (Section 2.4.5 [Files and Pipes], page 21.)
- Use sequences conveniently. (Section 2.6 [Sequences], page 24.)
- Show header fields in different fonts. (Section 3.1.1 [Customizing Viewing], page 31.)
- Find previously refiled messages. (Section 2.5 [Searching], page 22.)
- Place messages in a file. (Section 2.4.5 [Files and Pipes], page 21.)

Remember that you can also use MH commands when you're not running mh-e (and when you are!).

2 Using mh-e

This chapter leaves the tutorial style and goes into more detail about every mh-e command. The default, or "out of the box," behavior is documented. If this is not to your liking (for instance, you print with something other than `lpr`), see the associated section in Chapter 3 [Customizing mh-e], page 27, which is organized exactly like this chapter.

There are many commands, but don't get intimidated. There are command summaries at the beginning of each section. In case you have or would like to rebind the keys, the command summaries also list the associated Emacs Lisp function. Furthermore, even if you're stranded on a desert island with a laptop and are without your manuals, you can get a summary of all these commands with GNU Emacs online help: use *C-h m* (`describe-mode`) for a brief summary of commands or *C-h i* to read this manual via Info. The online help is quite good; try running *C-h C-h C-h*. This brings up a list of available help topics, one of which displays the documentation for a given key (like *C-h k C-n*). In addition, review Section 1.1 [Conventions], page 3, if any of the GNU Emacs conventions are strange to you.

Let's get started!

2.1 Reading Your Mail

The mh-e entry point for reading mail is *M-x mh-rmail*. This command incorporates your mail and creates a buffer called '`+inbox`' in MH-Folder mode. The *M-x mh-rmail* command shows you only new mail, not old mail.[1] The '`+inbox`' buffer contains *scan lines*, which are one-line summaries of each incorporated message. You can perform most MH commands on these messages via one-letter commands discussed in this chapter. See `scan`(1) for a description of the contents of the scan lines, and see the Figure in Section 1.4 [Reading Mail Tour], page 6, for an example.

RET	Display a message (`mh-show`).
SPC	Go to next page in message (`mh-page-msg`).
DEL	Go to previous page in message (`mh-previous-page`).
, (comma)	Display a message with all header fields (`mh-header-display`).
M-SPC	Go to next message in digest (`mh-page-digest`).
M-DEL	Go to previous message in digest (`mh-page-digest-backwards`).
M-b	Break up digest into separate messages (`mh-burst-digest`).
n	Display next message (`mh-next-undeleted-msg`).
p	Display previous message (`mh-previous-undeleted-msg`).
g	Go to a message (`mh-goto-msg`).
M-<	Go to first message (`mh-first-msg`).
M->	Go to last message (`mh-last-msg`).
t	Toggle between MH-Folder and MH-Folder Show modes (`mh-toggle-showing`).

[1] If you want to see your old mail as well, use *M-r* to pull all your messages into mh-e. Or, give a prefix argument to `mh-rmail` so it will prompt you for folder to visit like *M-f* (for example, *C-u M-x mh-rmail* `RET` *bob* `RET`). Both *M-r* and *M-f* are described in Section 2.4.3 [Organizing], page 21.

2.1.1 Viewing Your Mail

The *RET* (`mh-show`) command displays the message that the cursor is on. If the message is already displayed, it scrolls to the beginning of the message. Use `SPC` (`mh-page-msg`) and `DEL` (`mh-previous-page`) to move forwards and backwards one page at a time through the message. You can give either of these commands a prefix argument that specifies the number of lines to scroll (such as *10 SPC*). mh-e normally hides a lot of the superfluous header fields that mailers add to a message, but if you wish to see all of them, use the `,` (comma; `mh-header-display`) command.

2.1.1.1 Reading Digests

A digest is a message that contains other messages. Special mh-e commands let you read digests conveniently. You can use `SPC` and `DEL` to page through the digest as if it were a normal message, but if you wish to skip to the next message in the digest, use *M-SPC* (`mh-page-digest`). To return to a previous message, use *M-DEL* (`mh-page-digest-backwards`).

Another handy command is *M-b* (`mh-burst-digest`). This command uses the MH command `burst` to break out each message in the digest into its own message. Using this command, you can quickly delete unwanted messages, like this: Once the digest is split up, toggle out of MH-Folder Show mode with *t* (see Section 2.1.2 [Moving Around], page 12) so that the scan lines fill the screen and messages aren't displayed. Then use *d* (see Section 2.4.2 [Deleting], page 21) to quickly delete messages that you don't want to read (based on the '`Subject:`' header field). You can also burst the digest to reply directly to the people who posted the messages in the digest. One problem you may encounter is that the '`From:`' header fields are preceded with a '`>`' so that your reply can't create the '`To:`' field correctly. In this case, you must correct the '`To:`' field yourself. This is described later in Section 2.3.1 [Editing Textual], page 16.

2.1.1.2 Reading Multimedia Mail

MH has the ability to read MIME (Multipurpose Internet Mail Extensions) messages. Unfortunately, mh-e does not yet have this ability, so you have to use the MH commands `show` or `mhn` from the shell to read MIME messages.[2]

2.1.2 Moving Around

To move on to the next message, use the *n* (`mh-next-undeleted-msg`) command; use the *p* (`mh-previous-undeleted-msg`) command to read the previous message. Both of these commands can be given a prefix argument to specify how many messages to skip (for example, *5 n*). You can also move to a specific message with *g* (`mh-goto-msg`). You can enter the message number either before or after typing *g*. In the latter case, Emacs prompts you. Finally, you can go to the first or last message with *M-<* (`mh-first-msg`) and *M->* (`mh-last-msg`) respectively.

You can also use the Emacs commands *C-p* (`previous-line`) and *C-n* (`next-line`) to move up and down the scan lines in the MH-Folder window. These commands can be used in conjunction with *RET* to look at deleted or refiled messages.

[2] You can call them directly from Emacs if you're running the X Window System: type *M-! xterm -e mhn message-number*. You can leave out the `xterm -e` if you use `mhn -list` or `mhn -store`.

The command *t* (`mh-toggle-showing`) switches between MH-Folder mode and MH-Folder Show mode.[3] MH-Folder mode turns off the associated show buffer so that you can perform operations on the messages quickly without reading them. This is an excellent way to prune out your junk mail or to refile a group of messages to another folder for later examination.

2.2 Sending Mail

You can send a mail message in several ways. You can call *M-x mh-smail* directly, or from the command line like this:

```
% emacs -f mh-smail
```

From within mh-e's MH-Folder mode, other methods of sending mail are available as well:

m	Compose a message (`mh-send`).
r	Reply to a message (`mh-reply`).
f	Forward message(s) (`mh-forward`).
M-d	Redistribute a message (`mh-redistribute`).
M-e	Edit a message that was bounced by mailer (`mh-extract-rejected-mail`).
M-a	Edit a message to send it again (`mh-edit-again`).

From within a MH-Folder buffer, you can simply use the command *m* (`mh-send`). However you invoke `mh-send`, you are prompted for the '`To:`', '`cc:`', and '`Subject:`' header fields. Once you've specified the recipients and subject, your message appears in an Emacs buffer whose mode is MH-Letter (see the Figure in Section 2.2 [Sending Mail], page 13, to see what the buffer looks like). MH-Letter mode allows you to edit your message, to check the validity of the recipients, to insert other messages into your message, and to send the message. We'll go more into depth about editing a *draft*[4] (a message you're composing) in just a moment.

`mh-smail` always creates a two-window layout with the current buffer on top and the draft on the bottom. If you would rather preserve the window layout, use *M-x mh-smail-other-window*.

2.2.1 Replying to Mail

To compose a reply to a message, use the *r* (`mh-reply`) command. If you supply a prefix argument (as in *C-u r*), the message you are replying to is inserted in your reply after having first been run through `mhl` with the format file `mhl.reply`. See `mhl`(1) to see how you can modify the default `mhl.reply` file.

When you reply to a message, you are first prompted with '`Reply to whom?`'. You have several choices here.

[3] For you Emacs wizards, this is implemented as an Emacs minor mode.

[4] I highly recommend that you use a *draft folder* so that you can edit several drafts in parallel. To do so, create a folder (e.g., `+drafts`), and add a profile component called '`Draft-Folder:`' which contains `+drafts` (see `mh-profile`(5)).

Response	**Reply Goes To**
from	The person who sent the message. This is the default, so RET is sufficient.
to	Replies to the sender, plus all recipients in the 'To:' header field.
all *cc*	Forms a reply to the sender, plus all recipients.

Depending on your answer, `repl` is given a different argument to form your reply. Specifically, a choice of *from* or none at all runs `repl -nocc all`, and a choice of *to* runs `repl -cc to`. Finally, either *cc* or *all* runs `repl -cc all -nocc me`.

Two windows are then created. One window contains the message to which you are replying. Your draft, in MH-Letter mode (described in Section 2.3 [Draft Editing], page 15), is in the other window.

If you wish to customize the header or other parts of the reply draft, please see `repl`(1) and `mh-format`(5).

2.2.2 Forwarding Mail

To forward a message, use the *f* (`mh-forward`) command. You are given a draft to edit that looks like it would if you had run the MH command `forw`. You are given a chance to add some text (see Section 2.3 [Draft Editing], page 15).

You can forward several messages by using a prefix argument; in this case, you are prompted for the name of a *sequence*, a symbolic name that represents a list or range of message numbers (for example, *C-u f forbob* RET). All of the messages in the sequence are inserted into your draft. By the way, although sequences are often mentioned in this chapter, you don't have to worry about them for now; the full description of sequences in mh-e is at the end in Section 2.6 [Sequences], page 24. To learn more about sequences in general, please see `mh-sequence`(5).

2.2.3 Redistributing Your Mail

The command *M-d* (`mh-redistribute`) is similar in function to forwarding mail, but it does not allow you to edit the message, nor does it add your name to the '`From:`' header field. It appears to the recipient as if the message had come from the original sender. For more information on redistributing messages, see `dist`(1). Also investigate the *M-a* (`mh-edit-again`) command in Section 2.2.4 [Old Drafts], page 14, for another way to redistribute messages.

2.2.4 Editing Old Drafts and Bounced Messages

If you don't complete a draft for one reason or another, and if the draft buffer is no longer available, you can pick your draft up again with *M-a* (`mh-edit-again`). If you don't use a draft folder, your last `draft` file will be used. If you use draft folders, you'll need to visit the draft folder with *M-f drafts* RET, use *n* to move to the appropriate message, and then use *M-a* to prepare the message for editing.

The *M-a* command can also be used to take messages that were sent to you and to send them to more people.

Don't use *M-a* to re-edit a message from a *Mailer-Daemon* who complained that your mail wasn't posted for some reason or another. In this case, use *M-e* (`mh-extract-rejected-mail`) to prepare the message for editing by removing the *Mailer-Daemon* envelope and unneeded header fields. Fix whatever addressing problem you had, and send the message again with *C-c C-c*.

2.3 Editing a Draft

When you edit a message that you want to send (called a *draft* in this case), the mode used is MH-Letter. This mode provides several commands in addition to the normal Emacs editing commands to help you edit your draft.

C-c C-y	Insert contents of message to which you're replying (`mh-yank-cur-msg`).
C-c C-i	Insert a message from a folder (`mh-insert-letter`).
C-c C-f C-t	Move to '`To:`' header field (`mh-to-field`).
C-c C-f C-c	Move to '`cc:`' header field (`mh-to-field`).
C-c C-f C-s	Move to '`Subject:`' header field (`mh-to-field`).
C-c C-f C-f	Move to '`From:`' header field (`mh-to-field`).
C-c C-f C-b	Move to '`Bcc:`' header field (`mh-to-field`).
C-c C-f C-f	Move to '`Fcc:`' header field (`mh-to-fcc`).
C-c C-f C-d	Move to '`Dcc:`' header field (`mh-to-field`).
C-c C-w	Display expanded recipient list (`mh-check-whom`).
C-c C-s	Insert signature in message (`mh-insert-signature`).
C-c C-m C-f	Include forwarded message (MIME) (`mh-mhn-compose-forw`).
C-c C-m C-e	Include anonymous ftp reference (MIME) (`mh-mhn-compose-anon-ftp`).
C-c C-m C-t	Include anonymous ftp reference to compressed tar file (MIME) (`mh-mhn-compose-external-compressed-tar`).
C-c C-m C-i	Include binary, image, sound, etc. (MIME) (`mh-mhn-compose-insertion`).

C-c C-e	Run through `mhn` before sending (`mh-edit-mhn`).
C-c C-m C-u	Undo effects of `mhn` (`mh-revert-mhn-edit`).
C-c C-c	Save draft and send message (`mh-send-letter`).
C-c C-q	Quit editing and delete draft message (`mh-fully-kill-draft`).

2.3.1 Editing Textual Messages

The following sections show you how to edit a draft. The commands described here are also applicable to messages that have multimedia components.

2.3.1.1 Inserting letter to which you're replying

It is often useful to insert a snippet of text from a letter that someone mailed to provide some context for your reply. The command *C-c C-y* (`mh-yank-cur-msg`) does this by yanking a portion of text from the message to which you're replying and inserting '`> `' before each line.

You can control how much text is included when you run this command. If you run this command right away, without entering the buffer containing the message to you, this command will yank the entire message, as is, into your reply.[5] If you enter the buffer containing the message sent to you and move the cursor to a certain point and return to your reply and run *C-c C-y*, then the text yanked will range from that point to the end of the message. Finally, the most common action you'll perform is to enter the message sent to you, move the cursor to the beginning of a paragraph or phrase, set the *mark* with *C-SPC* or *C-@*, and move the cursor to the end of the paragraph or phrase. The cursor position is called the *point*, and the space between the mark and point is called the *region*. Having done that, *C-c C-y* will insert the region you selected.

2.3.1.2 Inserting messages

Messages can be inserted with *C-c C-i* (`mh-insert-letter`). This command prompts you for the folder and message number and inserts the message, indented by '`> `'. Certain undesirable header fields are removed before insertion. If given a prefix argument (like *C-u C-c C-i*), the header is left intact, the message is not indented, and '`> `' is not inserted before each line.

2.3.1.3 Editing the header

Because the header is part of the message, you can edit the header fields as you wish. However, several convenience functions exist to help you create and edit them. For example, the command *C-c C-f C-t* (`mh-to-field`; alternatively, *C-c C-f t*) moves the cursor to the '`To:`' header field, creating it if necessary. The functions to move to the '`cc:`', '`Subject:`', '`From:`', '`Bcc:`', and '`Dcc:`' header fields are similar.

One function behaves differently from the others, namely, *C-c C-f C-f* (`mh-to-fcc`; alternatively, *C-c C-f f*). This function will prompt you for the folder name in which to file a copy of the draft.

[5] If you'd rather have the header cleaned up, use *C-u r* instead of *r* when replying (see Section 2.2.1 [Replying], page 13).

Be sure to leave a row of dashes or a blank line between the header and the body of the message.

2.3.1.4 Checking recipients

The *C-c C-w* (`mh-check-whom`) command expands aliases so you can check the actual address(es) in the alias. A new buffer is created with the output of `whom`.

2.3.1.5 Inserting your signature

You can insert your signature at the current cursor location with the *C-c C-s* (`mh-insert-signature`) command. The text of your signature is taken from the file `~/.signature`.

2.3.2 Editing Multimedia Messages

mh-e has the capability to create multimedia messages. It uses the MIME (Multipurpose Internet Mail Extensions) protocol. The MIME protocol allows you to incorporate images, sound, video, binary files, and even commands that fetch a file with '`ftp`' when your recipient reads the message! If you were to create a multimedia message with plain MH commands, you would use `mhn`. Indeed, the mh-e MIME commands merely insert `mhn` directives which are later expanded by `mhn`.

Each of the mh-e commands for editing multimedia messages or for incorporating multimedia objects is prefixed with *C-c C-m* .

Several MIME objects are defined. They are called *content types*. The table in Section 3.3 [Customizing Draft Editing], page 35, contains a list of the content types that mh-e currently knows about. Several of the mh-e commands fill in the content type for you, whereas others require you to enter one. Most of the time, it should be obvious which one to use (e.g., use ***image/jpeg*** to include a JPEG image). If not, you can refer to RFC 1521,[6] which defines the MIME protocol, for a list of valid content types.

You are also sometimes asked for a *content description*. This is simply an optional brief phrase, in your own words, that describes the object. If you don't care to enter a content description, just press return and none will be included; however, a reader may skip over multimedia fields unless the content description is compelling.

Remember: you can always add `mhn` directives by hand.

2.3.2.1 Forwarding multimedia messages

Mail may be forwarded with MIME using the command *C-c C-m C-f* (`mh-mhn-compose-forw`). You are prompted for a content description, the name of the folder in which the messages to forward are located, and the messages' numbers.

2.3.2.2 Including an ftp reference

You can even have your message initiate an `ftp` transfer when the recipient reads the message. To do this, use the *C-c C-m C-e* (`mh-mhn-compose-anon-ftp`) command. You are prompted for the remote host and pathname, the content type, and the content description.

[6] This RFC (Request For Comments) is available via the URL `ftp://ds.internic.net/rfc/rfc1521.txt`.

2.3.2.3 Including tar files

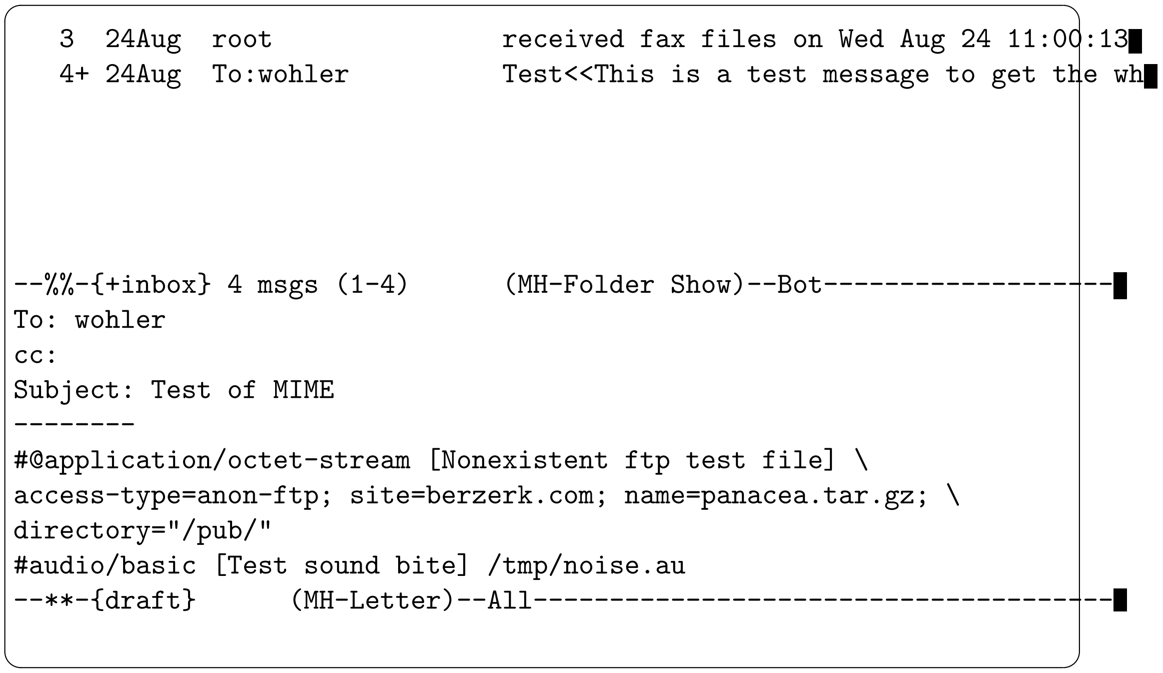
If the remote file (see Section 2.3.2.2 [FTP], page 17) is a compressed tar file, you can use *C-c C-m C-t* (`mh-mhn-compose-external-compressed-tar`). Then, in addition to retrieving the file via anonymous *ftp*, the file will also be uncompressed and untarred. You are prompted for the remote host and pathname and the content description. The pathname should contain at least one '`/`' (slash), because the pathname is broken up into directory and name components.

2.3.2.4 Including other multimedia objects

Images, sound, and video can be inserted in your message with the *C-c C-m C-i* (`mh-mhn-compose-insertion`) command. You are prompted for the filename containing the object, the content type, and a content description of the object.

2.3.2.5 Readying multimedia messages for sending

When you are finished editing a MIME message, it might look like this:



```
   3  24Aug  root              received fax files on Wed Aug 24 11:00:13
   4+ 24Aug  To:wohler         Test<<This is a test message to get the wh






--%%-{+inbox} 4 msgs (1-4)      (MH-Folder Show)--Bot------------------
To: wohler
cc:
Subject: Test of MIME
--------
#@application/octet-stream [Nonexistent ftp test file] \
access-type=anon-ftp; site=berzerk.com; name=panacea.tar.gz; \
directory="/pub/"
#audio/basic [Test sound bite] /tmp/noise.au
--**-{draft}      (MH-Letter)--All--------------------------------------
```

mh-e MIME *draft*

The lines added by the previous commands are `mhn` directives and need to be converted to MIME directives before sending. This is accomplished by the command *C-c C-e* (`mh-edit-mhn`), which runs `mhn` on the message. The following screen shows what those commands look like in full MIME format. You can see why mail user agents are usually built to hide these details from the user.

```
To: wohler
cc:
Subject: Test of MIME
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary="----- =_aaaaaaaaaa0"
Content-ID: <1623.777796162.0@newt.com>

------- =_aaaaaaaaaa0
Content-Type: message/external-body; access-type="anon-ftp";
        site="berzerk.com"; name="panacea.tar.gz"; directory="/pub/"

Content-Type: application/octet-stream
Content-ID: <1623.777796162.1@newt.com>
Content-Description: Nonexistent ftp test file

------- =_aaaaaaaaaa0
Content-Type: audio/basic
Content-ID: <1623.777796162.2@newt.com>
Content-Description: Test sound bite
Content-Transfer-Encoding: base64

Q3JlYXRpdmUgVm9pY2UgRmlsZRoaAAoBKREBQh8AgwCAgH9/f35+fn59fX5+fn5+f39/f39/f3
f4B/f39/f39/f39/f39/f39+f39+f39/f39/f4B/f39/fn5/f39/f3+Af39/f39/gH9/f39/fn
-----{draft}      (MH-Letter)--Top-------------------------------------
```

mh-e MIME *draft ready to send*

This action can be undone by running *C-c C-m C-u* (`mh-revert-mhn-edit`). It does this by reverting to a backup file. You are prompted to confirm this action, but you can avoid the confirmation by adding an argument (for example, *C-u C-c C-m C-u*).

2.3.3 Sending a Message

When you are all through editing a message, you send it with the *C-c C-c* (`mh-send-letter`) command. You can give an argument (as in *C-u C-c C-c*) to monitor the first stage of the delivery.

2.3.4 Killing the Draft

If for some reason you are not happy with the draft, you can kill it instead with *C-c C-q* (`mh-fully-kill-draft`). Emacs then kills the draft buffer and deletes the draft message.

2.4 Moving Your Mail Around

This section covers how messages and folders can be moved about or manipulated. Messages may be incorporated into your `+inbox`, deleted, and refiled. Messages containing `shar` or `uuencode` output can be stored. Folders can be visited, sorted, packed, or deleted. Here's a list of the available commands to do these things:

i	Incorporate new mail into folder (`mh-inc-folder`).
d	Delete message (`mh-delete-msg`).
C-d	Delete message, don't move to next message (`mh-delete-msg-no-motion`).
M-s	Find messages that meet search criteria (`mh-search-folder`).
o	Output (refile) message to folder (`mh-refile-msg`).
c	Copy message to folder (`mh-copy-msg`).
C-o	Output (write) message to file (`mh-write-msg-to-file`).
!	Repeat last output command (`mh-refile-or-write-again`).
l	Print message with `lpr` (`mh-print-msg`).
|	Pipe message through shell command (`mh-pipe-msg`).
M-n	Unpack message created with `uudecode` or `shar` (`mh-store-msg`).
M-l	List all folders (`mh-list-folders`).
M-f	Visit folder (`mh-visit-folder`).
M-r	Regenerate scan lines (`mh-rescan-folder`).
M-x mh-sort-folder	Sort folder.
M-p	Pack folder (`mh-pack-folder`).
M-k	Remove folder (`mh-kill-folder`).
x	Execute pending refiles and deletes (`mh-execute-commands`).
u	Undo pending refile or delete (`mh-undo`).
M-u	Undo all pending refiles and deletes (`mh-undo-folder`).
q	Quit (`mh-quit`).

2.4.1 Incorporating Your Mail

If at any time you receive new mail, incorporate the new mail into your '`+inbox`' buffer with *i* (`mh-inc-folder`). Note that *i* will display the '`+inbox`' buffer, even if there isn't any new mail. You can incorporate mail from any file into the current folder by specifying a prefix argument; you'll be prompted for the name of the file to use (for example, *C-u i ~/mbox* RET).

Emacs can notify you when you have new mail by displaying '`Mail`' in the mode line. To enable this behavior, and to have a clock in the mode line besides, add the following to `~/.emacs`:

```
(display-time)
```

2.4.2 Deleting Your Mail

To mark a message for deletion, use the *d* (`mh-delete-msg`) command. A '`D`' is placed by the message in the scan window, and the next message is displayed. If the previous command had been *p*, then the next message displayed is the message previous to the message just deleted. If you specify a prefix argument, you will be prompted for a sequence (see Section 2.6 [Sequences], page 24) to delete (for example, *C-u d frombob RET*). The *x* command actually carries out the deletion (see Section 2.4.6 [Finishing Up], page 22). *C-d* (`mh-delete-msg-no-motion`) marks the message for deletion but leaves the cursor at the current message in case you wish to perform other operations on the message.

2.4.3 Organizing Your Mail with Folders

mh-e has analogies for each of the MH `folder` and `refile` commands. To refile a message in another folder, use the *o* (`mh-refile-msg`) (mnemonic: "output") command. You are prompted for the folder name.

If you are refiling several messages into the same folder, you can use the *!* (`mh-refile-or-write-again`) command to repeat the last refile or write (see the description of *C-o* in Section 2.4.5 [Files and Pipes], page 21). Or, place the messages into a sequence (Section 2.6 [Sequences], page 24) and specify a prefix argument to *o*, in which case you'll be prompted for the name of the sequence (for example, *C-u o search RET*).

If you wish to copy a message to another folder, you can use the *c* (`mh-copy-msg`) command (see the `-link` argument to `refile`(1)). You are prompted for a folder, and you can specify a prefix argument if you want to copy a sequence into another folder. In this case, you are then prompted for the sequence. Note that unlike the *o* command, the copy takes place immediately. The original copy remains in the current folder.

When you want to read the messages that you have refiled into folders, use the *M-f* (`mh-visit-folder`) command to visit the folder. You are prompted for the folder name.

Other commands you can perform on folders include: *M-l* (`mh-list-folders`), to list all the folders in your mail directory; *M-k* (`mh-kill-folder`), to remove a folder; *M-x mh-sort-folder*, to sort the messages by date (see `sortm`(1) to see how to sort by other criteria); *M-p* (`mh-pack-folder`), to pack a folder, removing gaps from the numbering sequence; and *M-r* (`mh-rescan-folder`), to rescan the folder, which is useful to grab all messages in your `+inbox` after processing your new mail for the first time. If you don't want to rescan the entire folder, give *M-r* or *M-p* a prefix argument and you'll be prompted for a range of messages to display (for instance, *C-u M-r last:50 RET*).

2.4.4 Printing Your Mail

Printing mail is simple. Enter *l* (`mh-print-msg`) (for *l*ine printer or *l*pr). The message is formatted with `mhl` and printed with the `lpr` command. You can print all the messages in a sequence by specifying a prefix argument, in which case you are prompted for the name of the sequence (as in *C-u l frombob RET*).

2.4.5 Files and Pipes

mh-e does offer a couple of commands that are not a part of MH. The first one, *C-o* (`mh-write-msg-to-file`), writes a message to a file (think of the *o* as in "output"). You are prompted for the filename. If the file already exists, the message is appended to it. You can also write the message to the file without the header by specifying a prefix argument (such as *C-u C-o /tmp/foobar RET*). Subsequent writes to the same file can be made with the *!* command.

You can also pipe the message through a Unix shell command with the `|` (`mh-pipe-msg`) command. You are prompted for the Unix command through which you wish to run your message. If you give an argument to this command, the message header is included in the text passed to the command (the contrived example *C-u | lpr* would be done with the *l* command instead).

If the message is a shell archive `shar` or has been run through `uuencode` use *M-n* (`mh-store-msg`) to extract the body of the message. The default directory for extraction is the current directory, and you have a chance to specify a different extraction directory. The next time you use this command, the default directory is the last directory you used.

2.4.6 Finishing Up

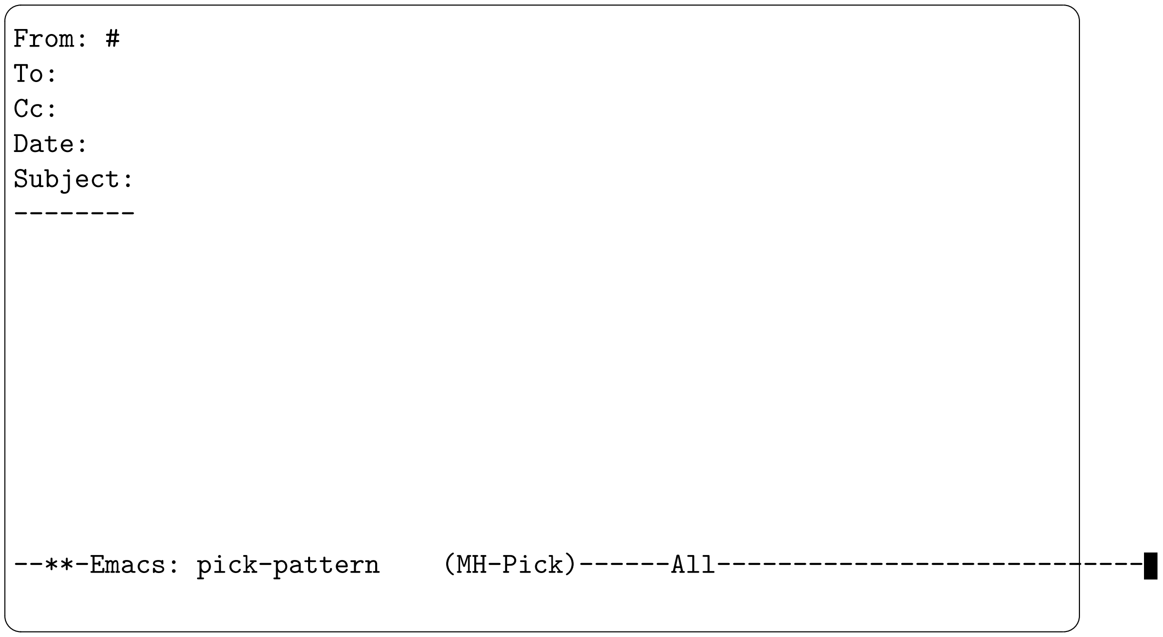
If you've deleted a message or refiled it, but changed your mind, you can cancel the action before you've executed it. Use *u* (`mh-undo`) to undo a refile on or deletion of a single message. You can also undo refiles and deletes for messages that belong to a given sequence by specifying a prefix argument. You'll be prompted for the name of the sequence (as in *C-u u frombob RET*). Alternatively, you can use *M-u* (`mh-undo-folder`) to undo all refiles or deletes in the current folder.

If you've marked messages to be deleted or refiled and you want to go ahead and delete or refile the messages, use *x* (`mh-execute-commands`). Many mh-e commands that may affect the numbering of the messages (such as *M-r* or *M-p*) will ask if you want to process refiles or deletes first and then either run *x* for you or undo the pending refiles and deletes, which are lost.

When you want to quit using mh-e and go back to editing, you can use the *q* (`mh-quit`) command. This buries the buffers of the current mh-e folder and restores the buffers that were present when you first ran *M-x mh-rmail*. You can later restore your mh-e session by selecting the '`+inbox`' buffer or by running *M-x mh-rmail* again.

2.5 Searching Through Messages

You can search a folder for messages to or from a particular person or about a particular subject. In fact, you can also search for messages containing selected strings in any arbitrary header field or any string found within the messages. Use the *M-s* (`mh-search-folder`) command. You are first prompted for the name of the folder to search and then placed in the following buffer in MH-Pick mode:



```
From: #
To:
Cc:
Date:
Subject:
--------










--**-Emacs: pick-pattern    (MH-Pick)------All----------------------------
```

Pick window

Edit this template by entering your search criteria in an appropriate header field that is already there, or create a new field yourself. If the string you're looking for could be anywhere in a message, then place the string underneath the row of dashes. The *`M-s`* command uses the MH command `pick` to do the real work, so read `pick`(1) to find out more about how to enter the criteria.

There are no semantics associated with the search criteria—they are simply treated as strings. Case is ignored when all lowercase is used, and regular expressions (a la `ed`) are available. It is all right to specify several search criteria. What happens then is that a logical *and* of the various fields is performed. If you prefer a logical *or* operation, run *`M-s`* multiple times.

As an example, let's say that we want to find messages from Ginnean about horseback riding in the Kosciusko National Park (Australia) during January, 1994. Normally we would start with a broad search and narrow it down if necessary to produce a manageable amount of data, but we'll cut to the chase and create a fairly restrictive set of criteria as follows:

```
From: ginnean
To:
Cc:
Date: Jan 1994
Subject: horse.*kosciusko
--------
```

As with MH-Letter mode, MH-Pick provides commands like *`C-c C-f C-t`* to help you fill in the blanks.

`C-c C-f C-t`
: Move to '`To:`' header field (`mh-to-field`).

C-c C-f C-c	Move to '`cc:`' header field (`mh-to-field`).
C-c C-f C-s	Move to '`Subject:`' header field (`mh-to-field`).
C-c C-f C-f	Move to '`From:`' header field (`mh-to-field`).
C-c C-f C-b	Move to '`Bcc:`' header field (`mh-to-field`).
C-c C-f C-f	Move to '`Fcc:`' header field (`mh-to-field`).
C-c C-f C-d	Move to '`Dcc:`' header field (`mh-to-field`).
C-c C-c	Execute the search (`mh-do-pick-search`).

To perform the search, type *C-c C-c* (`mh-do-pick-search`). The selected messages are placed in the *search* sequence, which you can use later in forwarding (see Section 2.2.2 [Forwarding], page 14), printing (see Section 2.4.4 [Printing], page 21), or narrowing your field of view (see Section 2.6 [Sequences], page 24). Subsequent searches are appended to the *search* sequence. If, however, you wish to start with a clean slate, first delete the *search* sequence (how to do this is discussed in Section 2.6 [Sequences], page 24).

If you're searching in a folder that is already displayed in a MH-Folder buffer, only those messages contained in the buffer are used for the search. Therefore, if you want to search in all messages, first kill the folder's buffer with *C-x k* or scan the entire folder with *M-r*.

2.6 Using Sequences

For the whole scoop on MH sequences, refer to `mh-sequence`(5). As you've read, several of the mh-e commands can operate on a sequence, which is a shorthand for a range or group of messages. For example, you might want to forward several messages to a friend or colleague. Here's how to manipulate sequences.

%	Put message in a sequence (`mh-put-msg-in-seq`).
?	Display sequences that message belongs to (`mh-msg-is-in-seq`).
M-q	List all sequences in folder (`mh-list-sequences`).
M-%	Remove message from sequence (`mh-delete-msg-from-seq`).
M-#	Delete sequence (`mh-delete-seq`).
C-x n	Restrict display to messages in sequence (`mh-narrow-to-seq`).
C-x w	Remove restriction; display all messages (`mh-widen`).
M-x mh-update-sequences	Push mh-e's state out to MH.

To place a message in a sequence, use *%* (`mh-put-msg-in-seq`) to do it manually, or use the MH command `pick` or the mh-e version of `pick` (Section 2.5 [Searching], page 22) which create a sequence automatically. Give *%* a prefix argument and you can add all the messages in one sequence to another sequence (for example, *C-u % SourceSequence RET*).

Once you've placed some messages in a sequence, you may wish to narrow the field of view to just those messages in the sequence you've created. To do this, use *C-x n* (`mh-narrow-to-seq`). You are prompted for the name of the sequence. What this does is show only those messages that are in the selected sequence in the MH-Folder buffer. In addition, it limits further mh-e searches to just those messages. When you want to widen the view to all your messages again, use *C-x w* (`mh-widen`).

You can see which sequences a message is in with the *?* (`mh-msg-is-in-seq`) command. Or, you can list all sequences in a selected folder (default is current folder) with *M-q* (`mh-list-sequences`).

If you want to remove a message from a sequence, use *M-%* (`mh-delete-msg-from-seq`), and if you want to delete an entire sequence, use *M-#* (`mh-delete-seq`). In the latter case you are prompted for the sequence to delete. Note that this deletes only the sequence, not the messages in the sequence. If you want to delete the messages, use *C-u d* (see Section 2.4.2 [Deleting], page 21, above).

Two sequences are maintained internally by mh-e and pushed out to MH when you type either the *x* or *q* command. They are the sequence specified by your '`Unseen-Sequence:`' profile entry and *cur*. However, you can also just update MH's state with the command *M-x mh-update-sequences*. See Section 3.1.1 [Customizing Viewing], page 31, for an example of how this command might be used.

With the exceptions of *C-x n* and *C-x w*, the underlying MH command dealing with sequences is `mark`.

2.7 Miscellaneous Commands

One other command worth noting is *M-x mh-version*. Since there were a few changes in command letters between Versions 3 and 4, use this command to see which version you are running. This command didn't exist before Version 4, so the message '`[No match]`' indicates that it's time to upgrade (see Section A.4 [Getting mh-e], page 45). In the meantime, use the older commands that are listed in Appendix C [Changes to mh-e], page 49. The output of *M-x mh-version* should also be included with any bug report you send (see Section A.1 [Bug Reports], page 45).

3 Customizing mh-e

Until now, we've talked about the mh-e commands as they work "out of the box." Of course, it is also possible to reconfigure mh-e beyond recognition. The following sections describe all of the customization variables, show the defaults, and make recommendations for customization. The outline of this chapter is identical to that of Chapter 2 [Using mh-e], page 11, to make it easier to find the variables you'd need to modify to affect a particular command.

However, when customizing your mail environment, first try to change what you want in MH, and only change mh-e if changing MH is not possible. That way you will get the same behavior inside and outside GNU Emacs. Note that mh-e does not provide hooks for customizations that can be done in MH; this omission is intentional.

Many string or integer variables are easy enough to modify using Emacs Lisp. Any such modifications should be placed in a file called `.emacs` in your home directory (that is, `~/.emacs`). For example, to modify the variable that controls printing, you could add:

```
(setq mh-lpr-command-format "nenscript -G -r -2 -i'%s'")
```

Section 3.4.4 [Customizing Printing], page 42, talks more about this variable.

Variables can also hold Boolean values. In Emacs Lisp, the Boolean values are `nil`, which means false, and `t`, which means true. Usually, variables are turned off by setting their value to `nil`, as in

```
(setq mh-bury-show-buffer nil)
```

which keeps the MH-Show buffer at the top of the buffer stack. To turn a variable on, you use

```
(setq mh-bury-show-buffer t)
```

which places the MH-Show buffer at the bottom of the buffer stack. However, the text says to turn on a variable by setting it to a *non*-`nil` value, because sometimes values other than `t` are meaningful (for example, see `mhl-formfile`, described in Section 3.1.1 [Customizing Viewing], page 31). Other variables, such as hooks, involve a little more Emacs Lisp programming expertise.

You can also "preview" the effects of changing variables before committing the changes to `~/.emacs`. Variables can be changed in the current Emacs session by using *`M-x set-variable`*.

In general, *commands* in this text refer to Emacs Lisp functions. Programs outside of Emacs are specifically called MH commands, shell commands, or Unix commands.

I hope I've included enough examples here to get you well on your way. If you want to explore Emacs Lisp further, a programming manual does exist,[1] and you can look at the code itself for examples. Look in the Emacs Lisp directory on your system (such as `/usr/local/lib/emacs/lisp`) and find all the `mh-*.el` files there. When calling mh-e and other Emacs Lisp functions directly from Emacs Lisp code, you'll need to know the correct arguments. Use the online help for this. For example, try *`C-h f mh-execute-commands`*

[1] The *GNU Emacs Lisp Reference Manual* may be available online in the Info system by typing *`C-h i m Emacs Lisp RET`*. If not, you can order a printed manual, which has the desirable side-effect of helping to support the Free Software Foundation which made all this great software available. You can find an order form by running *`C-h C-d`*, or you can request an order form from *gnu@prep.ai.mit.edu*.

RET. If you write your own functions, please do not prefix your symbols (variables and functions) with `mh-`. This prefix is reserved for the mh-e package. To avoid conflicts with existing mh-e symbols, use a prefix like `my-` or your initials.

3.1 Reading Your Mail

I'll start out by including a function that I use as a front end to mh-e.[2] It toggles between your working window configuration, which may be quite involved—windows filled with source, compilation output, man pages, and other documentation—and your mh-e window configuration. Like the rest of the customization described in this chapter, simply add the following code to `~/.emacs`. Don't be intimidated by the size of this example; most customizations are only one line.

[2] Stephen Gildea's favorite binding is *(global-set-key "\C-cr" 'mh-rmail)*.

```
Starting mh-e

(defvar my-mh-screen-saved nil
  "Set to non-nil when mh-e window configuration shown.")
(defvar my-normal-screen nil "Normal window configuration.")
(defvar my-mh-screen nil "mh-e window configuration.")

(defun my-mh-rmail (&optional arg)
  "Toggle between mh-e and normal screen configurations.
With non-nil or prefix argument, inc mailbox as well
when going into mail."
  (interactive "P")                    ; user callable function, P=prefix arg
  (setq my-mh-screen-saved             ; save state
        (cond
         ;; Bring up mh-e screen if arg or normal window configuration.
         ;; If arg or +inbox buffer doesn't exist, run mh-rmail.
         ((or arg (null my-mh-screen-saved))
          (setq my-normal-screen (current-window-configuration))
          (if (or arg (null (get-buffer "+inbox")))
              (mh-rmail)
            (set-window-configuration my-mh-screen))
          t)                           ; set my-mh-screen-saved to t
         ;; Otherwise, save mh-e screen and restore normal screen.
         (t
          (setq my-mh-screen (current-window-configuration))
          (set-window-configuration my-normal-screen)
          nil))))                      ; set my-mh-screen-saved to nil

(global-set-key "\C-x\r" 'my-mh-rmail)  ; call with C-x RET
```

If you type an argument (*C-u*) or if `my-mh-screen-saved` is `nil` (meaning a non-mh-e window configuration), the current window configuration is saved, either +inbox is displayed or `mh-rmail` is run, and the mh-e window configuration is shown. Otherwise, the mh-e window configuration is saved and the original configuration is displayed.

Now to configure mh-e. The following table lists general mh-e variables and variables that are used while reading mail.

`mh-progs` Directory containing MH programs (default: dynamic).

`mh-lib` Directory containing MH support files and programs (default: dynamic).

`mh-do-not-confirm`
: Don't confirm on non-reversible commands (default: `nil`).

`mh-summary-height`
: Number of scan lines to show (includes mode line) (default: 4).

`mh-folder-mode-hook`
: Functions to run in MH-Folder mode (default: `nil`).

`mh-clean-message-header`
: Remove extraneous headers (default: `nil`).

`mh-invisible-headers`
: Headers to hide (default: ‘`"^Received: \\| ^Message-Id: \\| ^Remailed-\\| ^Via: \\| ^Mail-from: \\| ^Return-Path: \\| ^In-Reply-To: \\| ^Resent-"`’).

`mh-visible-headers`
: Headers to display (default: `nil`).

`mhl-formfile`
: Format file for `mhl` (default: `nil`).

`mh-show-hook`
: Functions to run when showing message (default: `nil`).

`mh-show-mode-hook`
: Functions to run when showing message (default: `nil`).

`mh-bury-show-buffer`
: Leave show buffer at bottom of stack (default: `t`).

`mh-show-buffer-mode-line-buffer-id`
: Name of show buffer in mode line (default: ‘`"{show-%s} %d"`’).

The two variables `mh-progs` and `mh-lib` are used to tell mh-e where the MH programs and supporting files are kept, respectively. mh-e does try to figure out where they are kept for itself by looking in common places and in the user's ‘`PATH`’ environment variable, but if it cannot find the directories, or finds the wrong ones, you should set these variables. The name of the directory should be placed in double quotes, and there should be a trailing slash (‘`/`’). See the example in Section 1.2 [Getting Started], page 4.

If you never make mistakes, and you do not like confirmations for your actions, you can set `mh-do-not-confirm` to a non-`nil` value to disable confirmation for unrecoverable commands such as *M-k* (`mh-kill-folder`) and *M-u* (`mh-undo-folder`). Here's how you set boolean values:

```
(setq mh-do-not-confirm t)
```

The variable `mh-summary-height` controls the number of scan lines displayed in the MH-Folder window, including the mode line. The default value of 4 means that 3 scan lines are displayed. Here's how you set numerical values:

```
(setq mh-summary-height 2)              ; only show the current scan line
```

Normally the buffer for displaying messages is buried at the bottom at the buffer stack. You may wish to disable this feature by setting `mh-bury-show-buffer` to `nil`. One advantage of not burying the show buffer is that one can delete the show buffer more easily in an electric buffer list because of its proximity to its associated MH-Folder buffer. Try running *`M-x electric-buffer-list`* to see what I mean.

The hook `mh-folder-mode-hook` is called when a new folder is created with MH-Folder mode. This could be used to set your own key bindings, for example:

```
Create additional key bindings via mh-folder-mode-hook

(defvar my-mh-init-done nil "Non-nil when one-time mh-e settings made.")

(defun my-mh-folder-mode-hook ()
  "Hook to set key bindings in MH-Folder mode."
  (if (not my-mh-init-done)             ; only need to bind the keys once
      (progn
        (local-set-key "/" 'search-msg)
        (local-set-key "b" 'mh-burst-digest)    ; better use of b
        (setq my-mh-init-done t))))

;;; Emacs 19
(add-hook 'mh-folder-mode-hook 'my-mh-folder-mode-hook)
;;; Emacs 18
;;;   (setq mh-folder-mode-hook (cons 'my-mh-folder-mode-hook
;;;                                   mh-folder-mode-hook))

(defun search-msg ()
  "Search for a regexp in the current message."
  (interactive)                         ; user function
  (save-window-excursion
    (other-window 1)                    ; go to next window
    (isearch-forward-regexp)))          ; string search; hit return (ESC
                                        ;   in Emacs 18) when done
```

3.1.1 Viewing Your Mail

Several variables control what displayed messages look like. Normally messages are delivered with a handful of uninteresting header fields. You can make them go away by setting `mh-clean-message-header` to a non-`nil` value. The header can then be cleaned up in two ways. By default, the header fields in `mh-invisible-headers` are removed. On the other hand, you could set `mh-visible-headers` to the fields that you would like to see. If this variable is set, `mh-invisible-headers` is ignored. I suggest that you not set `mh-visible-headers` since if you use this variable, you might miss a lot of header fields that you'd rather not miss. As an example of how to set a string variable, `mh-visible-headers` can be set to show a minimum set of header fields (see (Section "Syntax of Regular Expressions" in *The GNU Emacs Manual*, for a description of the special characters in this string):

```
(setq mh-visible-headers "^From: \\|^Subject: \\|^Date: ")
```

Normally mh-e takes care of displaying messages itself (rather than calling an MH program to do the work). If you'd rather have `mhl` display the message (within mh-e), set the variable `mhl-formfile` to a non-`nil` value. You can set this variable either to `t` to use the default format file or to a filename if you have your own format file (`mhl`(1) tells you how to write one). When writing your own format file, use a nonzero value for `overflowoffset` to ensure the header is RFC 822 compliant and parseable by mh-e. `mhl` is always used for printing and forwarding; in this case, the value of `mhl-formfile` is consulted if it is a filename.

Two hooks can be used to control how messages are displayed. The first hook, `mh-show-mode-hook`, is called early on in the process of displaying of messages. It is used to perform some actions on the contents of messages, such as highlighting the header fields. If you're running Emacs 19 under the X Window System, the following example will highlight the '`From:`' and '`Subject:`' header fields. This is a very nice feature indeed.

```
Emphasize header fields in different fonts via mh-show-mode-hook

(defvar my-mh-keywords
   '(("^From: \\(.*\\)" 1 'bold t)
     ("^Subject: \\(.*\\)" 1 'highlight t))
  "mh-e additions for font-lock-keywords.")

(defun my-mh-show-mode-hook ()
  "Hook to turn on and customize fonts."
  (require 'font-lock)                 ; for font-lock-keywords below
  (make-local-variable 'font-lock-mode-hook) ; don't affect other buffers
  (add-hook 'font-lock-mode-hook       ; set a hook with inline function
            (function                  ; modifies font-lock-keywords when
             (lambda ()                ; font-lock-mode run
               (setq font-lock-keywords
                     (append my-mh-keywords font-lock-keywords)))))
  (font-lock-mode 1))                  ; change the typefaces

(if window-system                      ; can't do this on ASCII terminal
    (add-hook 'mh-show-mode-hook 'my-mh-show-mode-hook))
```

The second hook, `mh-show-hook`, is the last thing called after messages are displayed. It's used to affect the behavior of mh-e in general or when `mh-show-mode-hook` is too early. For example, if you wanted to keep mh-e in sync with MH, you could use `mh-show-hook` as follows:

```
(add-hook 'mh-show-hook 'mh-update-sequences)
```

The function `mh-update-sequences` is documented in Section 2.4.6 [Finishing Up], page 22. For those who like to modify their mode lines, use `mh-show-buffer-mode-line-buffer-id` to modify the mode line in the MH-Show buffers. Place the two escape strings '`%s`' and '`%d`', which will display the folder name and the message number, respectively, somewhere in the string in that order. The default value of ‘`"{show-%s} %d"`’ yields a mode line of

```
-----{show-+inbox} 4      (MH-Show)--Bot----------------------------------
```

3.1.2 Moving Around

When you use *t* (`mh-toggle-showing`) to toggle between show mode and scan mode, the MH-Show buffer is hidden and the MH-Folder buffer is left alone. Setting `mh-recenter-summary-p` to a non-`nil` value causes the toggle to display as many scan lines as possible, with the cursor at the middle. The effect of `mh-recenter-summary-p` is rather useful, but it can be annoying on a slow network connection.

3.2 Sending Mail

You may wish to start off by adding the following useful key bindings to your `.emacs` file:

```
(global-set-key "\C-xm" 'mh-smail)
(global-set-key "\C-x4m" 'mh-smail-other-window)
```

In addition, several variables are useful when sending mail or replying to mail. They are summarized in the following table.

`mh-comp-formfile`	Format file for drafts (default: ‘`"components"`’).
`mh-repl-formfile`	Format file for replies (default: ‘`"replcomps"`’).
`mh-letter-mode-hook`	Functions to run in MH-Letter mode (default: `nil`).
`mh-compose-letter-function`	Functions to run when starting a new draft (default: `nil`).
`mh-reply-default-reply-to`	Whom reply goes to (default: `nil`).
`mh-forward-subject-format`	Format string for forwarded message subject (default: ‘`"%s: %s"`’).
`mh-redist-full-contents`	`send` requires entire message (default: `nil`).
`mh-new-draft-cleaned-headers`	Remove these header fields from re-edited draft (default: ‘`"^Date:\\| ^Received:\\| ^Message-Id:\\| ^From:\\| ^Sender:\\| ^Delivery-Date:\\| ^Return-Path:"`’).

Since mh-e does not use `comp` to create the initial draft, you need to set `mh-comp-formfile` to the name of your components file if it isn’t `components`. This is the name of the file that contains the form for composing messages. If it does not contain an absolute pathname, mh-e searches for the file first in your MH directory and then in the system MH library directory (such as `/usr/local/lib/mh`). Replies, on the other hand, are built using `repl`. You can change the location of the field file from the default of `replcomps` by modifying `mh-repl-formfile`.

Two hooks are provided to run commands on your freshly created draft. The first hook, `mh-letter-mode-hook`, allows you to do some processing before editing a letter. For example, you may wish to modify the header after `repl` has done its work, or you may have a complicated `components` file and need to tell mh-e where the cursor should go. Here's an example of how you would use this hook—all of the other hooks are set in this fashion as well.

Prepare draft for editing via mh-letter-mode-hook

```
(defvar letter-mode-init-done nil
  "Non-nil when one-time mh-e settings have made.")

(defun my-mh-letter-mode-hook ()
  "Hook to prepare letter for editing."
  (if (not letter-mode-init-done)    ; only need to bind the keys once
      (progn
        (local-set-key "\C-ctb" 'add-enriched-text)
        (local-set-key "\C-cti" 'add-enriched-text)
        (local-set-key "\C-ctf" 'add-enriched-text)
        (local-set-key "\C-cts" 'add-enriched-text)
        (local-set-key "\C-ctB" 'add-enriched-text)
        (local-set-key "\C-ctu" 'add-enriched-text)
        (local-set-key "\C-ctc" 'add-enriched-text)
        (setq letter-mode-init-done t)))
  (setq fill-prefix "  ")            ; I find indented text easier to read
  (save-excursion
    (goto-char (point-max))          ; go to end of message to
    (mh-insert-signature)))          ;   insert signature

(add-hook 'mh-letter-mode-hook 'my-mh-letter-mode-hook)
```

The function, `add-enriched-text` is defined in the example in Section 3.3.2 [Customizing Editing MIME], page 37.

The second hook, a function really, is `mh-compose-letter-function`. Like `mh-letter-mode-hook`, it is called just before editing a new message; however, it is the last function called before you edit your message. The consequence of this is that you can write a function to write and send the message for you. This function is passed three arguments: the contents of the '`To:`', '`Subject:`', and '`cc:`' header fields.

3.2.1 Replying to Mail

If you find that most of the time that you specify *cc* when you reply to a message, set `mh-reply-default-reply-to` to '`cc`'. This variable is normally set to `nil` so that you are prompted for the recipient of a reply. It can be set to one of '`from`', '`to`', or '`cc`'; you are then no longer prompted for the recipient(s) of your reply.

3.2.2 Forwarding Mail

When forwarding a message, the format of the '`Subject:`' header field can be modified by the variable `mh-forward-subject-format`. This variable is a string which includes two escapes (‘`%s`’). The first ‘`%s`’ is replaced with the sender of the original message, and the second one is replaced with the original ‘`Subject:`’. The default value of ‘`"%s: %s"`’ takes a message with the header:

```
To: Bill Wohler <wohler@newt.com>
Subject: Re: 49er football
From: Greg DesBrisay <gd@cellnet.com>
```

and creates a subject header field of:

```
Subject: Greg DesBrisay: Re: 49er football
```

3.2.3 Redistributing Your Mail

The variable `mh-redist-full-contents` must be set to non-`nil` if `dist` requires the whole letter for redistribution, which is the case if `send` is compiled with the BERK[3] option (which many people abhor). If you find that MH will not allow you to redistribute a message that has been redistributed before, this variable should be set to `nil`.

3.2.4 Editing Old Drafts and Bounced Messages

The header fields specified by `mh-new-draft-cleaned-headers` are removed from an old draft that has been recreated with *`M-e`* (`mh-extract-rejected-mail`) or *`M-a`* (`mh-edit-again`). If when you edit an old draft with these commands you find that there are header fields that you don’t want included, you can append them to this variable. For example,

```
(setq mh-new-draft-cleaned-headers
      (concat mh-new-draft-cleaned-headers "\\|^Some-Field:"))
```

This appends the regular expression ‘`\\|^Some-Field:`’ to the variable (see Section “Syntax of Regular Expressions” in *The GNU Emacs Manual*). The ‘`\\|`’ means *or*, and the ‘`^`’ (caret) matches the beginning of the line. This is done to be very specific about which fields match. The literal ‘`:`’ is appended for the same reason.

3.3 Editing a Draft

There are several variables used during the draft editing phase. Examples include changing the name of the file that holds your signature or telling mh-e about new multimedia types. They are:

`mh-yank-from-start-of-msg`
: How to yank when region not set (default: `t`).

`mh-ins-buf-prefix`
: Indent for yanked messages (default: ‘`"> "`’).

`mail-citation-hook`
: Functions to run on yanked messages (default: `nil`).

`mh-delete-yanked-msg-window`
: Delete message window on yank (default: `nil`).

[3] To see which options your copy of MH was compiled with, use *`M-x mh-version`* (Section 2.7 [Miscellaneous], page 25).

`mh-mime-content-types`
: List of valid content types (default: ‘`'(("text/plain") ("text/richtext") ("multipart/mixed") ("multipart/alternative") ("multipart/digest") ("multipart/parallel") ("message/rfc822") ("message/partial") ("message/external-body") ("application/octet-stream") ("application/postscript") ("image/jpeg") ("image/gif") ("audio/basic") ("video/mpeg"))`’).

`mh-mhn-args`
: Additional arguments for `mhn` (default: `nil`).

`mh-signature-file-name`
: File containing signature (default: ‘`"~/.signature"`’).

`mh-before-send-letter-hook`
: Functions to run before sending draft (default: `nil`).

`mh-send-prog`
: MH program used to send messages (default: ‘`"send"`’).

3.3.1 Editing Textual Messages

The following two sections include variables that customize the way you edit a draft. The discussion here applies to editing multimedia messages as well.

3.3.1.1 Inserting letter to which you're replying

To control how much of the message to which you are replying is yanked by *`C-c C-y`* (`mh-yank-cur-msg`) into your reply, modify `mh-yank-from-start-of-msg`. The default value of `t` means that the entire message is copied. If it is set to `'body` (don't forget the apostrophe), then only the message body is copied. If it is set to `nil`, only the part of the message following point (the current cursor position in the message's buffer) is copied. In any case, this variable is ignored if a region is set in the message you are replying to. The string contained in `mh-ins-buf-prefix` is inserted before each line of a message that is inserted into a draft with *`C-c C-y`* (`mh-yank-cur-msg`). I suggest that you not modify this variable. The default value of ‘`"> "`’ is the default string for many mailers and news readers: messages are far easier to read if several included messages have all been indented by the same string. The variable `mail-citation-hook` is `nil` by default, which means that when a message is inserted into the letter, each line is prefixed by `mh-ins-buf-prefix`. Otherwise, it can be set to a function that modifies an included citation.[4] If you like to yank all the text from the message you're replying to in one go, set `mh-delete-yanked-msg-window` to non-`nil` to delete the window containing the original message after yanking it to make more room on your screen for your reply.

3.3.1.2 Inserting your signature

You can change the name of the file inserted with *`C-c C-s`* (`mh-insert-signature`) by changing `mh-signature-file-name` (default: `"~/.signature"`).

[4] *Supercite* is an example of a full-bodied, full-featured citation package. It is in Emacs versions 19.15 and later, and can be found via anonymous `ftp` on ‘`archive.cis.ohio-state.edu`’ in `/pub/gnu/emacs/elisp-archive/packages/sc3.1.tar.Z`

3.3.2 Editing Multimedia Messages

The variable `mh-mime-content-types` contains a list of the currently valid content types. They are listed in the table in Section 3.3 [Customizing Draft Editing], page 35. If you encounter a new content type, you can add it like this:

```
(setq mh-mime-content-types (append mh-mime-content-types
                                    '(("new/type"))))
```

Emacs macros can be used to insert enriched text directives like '`<bold>`'. The following code will make, for example, *C-c t b* insert the '`<bold>`' directive.

```
Emacs macros for entering enriched text

(defvar enriched-text-types '(("b" . "bold") ("i" . "italic") ("f" . "fixed")
                              ("s" . "smaller") ("B" . "bigger")
                              ("u" . "underline") ("c" . "center"))
  "Alist of (final-character . directive) choices for add-enriched-text.
Additional types can be found in RFC 1563.")

(defun add-enriched-text (begin end)
  "Add enriched text directives around region.
The directive used comes from the list enriched-text-types and is
specified by the last keystroke of the command.  When called from Lisp,
arguments are BEGIN and END."
  (interactive "r")
  ;; Set type to the directive indicated by the last keystroke.
  (let ((type (cdr (assoc (char-to-string (logior last-input-char ?`))
                          enriched-text-types))))
    (save-restriction              ; restores state from narrow-to-region
      (narrow-to-region begin end) ; narrow view to region
      (goto-char (point-min))      ; move to beginning of text
      (insert "<" type ">")        ; insert beginning directive
      (goto-char (point-max))      ; move to end of text
      (insert "</" type ">"))))    ; insert terminating directive
```

To use the function `add-enriched-text`, first create keybindings for it (see Section 3.2 [Customizing Sending], page 33). Then, set the mark with *C-@* or *C-SPC*, type in the text to be highlighted, and type *C-c t b*. This adds '`<bold>`' where you set the mark and adds '`</bold>`' at the location of your cursor, giving you something like: '`You should be <bold>very</bold>`'. You may also be interested in investigating `sgml-mode`.

3.3.2.1 Readying multimedia messages for sending

If you wish to pass additional arguments to `mhn` to affect how it builds your message, use the variable `mh-mhn-args`. For example, you can build a consistency check into the message by setting `mh-mhn-args` to `-check`. The recipient of your message can then run `mhn -check` on the message—`mhn` will complain if the message has been corrupted on the way. The *C-c C-e* (`mh-mhn-edit`) command only consults this variable when given a prefix argument.

3.3.3 Sending a Message

If you want to check your spelling in your message before sending, use `mh-before-send-letter-hook` like this:

Spell-check message via mh-before-send-letter-hook

```
(add-hook 'mh-before-send-letter-hook 'ispell-message)
```

In case the MH `send` program is installed under a different name, use `mh-send-prog` to tell mh-e the name.

3.4 Moving Your Mail Around

If you change the name of some of the MH programs or have your own printing programs, the following variables can help you. They are described in detail in the subsequent sections.

`mh-inc-prog`
: Program to incorporate mail (default: ‘`"inc"`’).

`mh-inc-folder-hook`
: Functions to run when incorporating mail (default: `nil`).

`mh-delete-msg-hook`
: Functions to run when deleting messages (default: `nil`).

`mh-print-background`
: Print in foreground or background (default: `nil`).

`mh-lpr-command-format`
: Command used to print (default: ‘`"lpr -J '%s'"`’).

`mh-default-folder-for-message-function`
: Function to generate a default folder (default: `nil`).

`mh-auto-folder-collect`
: Collect folder names in background at startup (default: `t`).

`mh-recursive-folders`
: Collect nested folders (default: `nil`).

`mh-refile-msg-hook`
: Functions to run when refiling message (default: `nil`).

`mh-store-default-directory`
: Default directory for storing files created by `uuencode` or `shar` (default: `nil`).

`mh-sortm-args`
: Additional arguments for `sortm` (default: `nil`).

`mh-scan-prog`
: Program to scan messages (default: ‘`"scan"`’).

`mh-before-quit-hook`
: Functions to run before quitting (default: `nil`). See also `mh-quit-hook`.

`mh-quit-hook`
: Functions to run after quitting (default: `nil`). See also `mh-before-quit-hook`.

3.4.1 Incorporating Your Mail

The name of the program that incorporates new mail is stored in `mh-inc-prog`; it is '`"inc"`' by default. This program generates a one-line summary for each of the new messages. Unless it is an absolute pathname, the file is assumed to be in the `mh-progs` directory. You may also link a file to `inc` that uses a different format (see `mh-profile`(5)). You'll then need to modify several variables appropriately; see `mh-scan-prog` below. You can set the hook `mh-inc-folder-hook`, which is called after new mail is incorporated by the *`i`* (`mh-inc-folder`) command. A good use of this hook is to rescan the whole folder either after running *`M-x mh-rmail`* the first time or when you've changed the message numbers from outside of mh-e.

`Rescan folder after incorporating new mail via mh-inc-folder-hook`

```
(defun my-mh-inc-folder-hook ()
  "Hook to rescan folder after incorporating mail."
  (if (buffer-modified-p)            ; if outstanding refiles and deletes,
      (mh-execute-commands))         ;   carry them out
  (mh-rescan-folder)                 ; synchronize with +inbox
  (mh-show))                         ; show the current message

(add-hook 'mh-inc-folder-hook 'my-mh-inc-folder-hook)
```

3.4.2 Deleting Your Mail

The hook `mh-delete-msg-hook` is called after you mark a message for deletion. For example, the current maintainer of mh-e used this once when he kept statistics on his mail usage.

3.4.3 Organizing Your Mail with Folders

By default, operations on folders work only one level at a time. Set `mh-recursive-folders` to non-`nil` to operate on all folders. This mostly means that you'll be able to see all your folders when you press `TAB` when prompted for a folder name. The variable `mh-auto-folder-collect` is normally turned on to generate a list of folder names in the background as soon as mh-e is loaded. Otherwise, the list is generated when you need a folder name the first time (as with *`o`* (`mh-refile-msg`)). If you have a lot of folders and you have `mh-recursive-folders` set, this could take a while, which is why it's nice to do the folder collection in the background.

The function `mh-default-folder-for-message-function` is used by *`o`* (`mh-refile-msg`) and *`C-c C-f C-f`* (`mh-to-fcc`) to generate a default folder. The generated folder name should be a string with a '`+`' before it. For each of my correspondents, I use the same name for both an alias and a folder. So, I wrote a function that takes the address in the '`From:`' header field, finds it in my alias file, and returns the alias, which is used as a default folder name. This is the most complicated example given here, and it demonstrates several features of Emacs Lisp programming. You should be able to drop this into `~/.emacs`, however. If you use this to store messages in a subfolder of your Mail directory, you can modify the line that starts '`(format +%s...`' and insert your subfolder after the folder symbol '`+`'.

Creating useful default folder for refiling via mh-default-folder-for-message-function

```
(defun my-mh-folder-from-address ()
  "Determine folder name from address.
Takes the address in the From: header field, and returns its corresponding
alias from the user's personal aliases file. Returns nil if the address
was not found."
  (require 'rfc822)                         ; for the rfc822 functions
  (search-forward-regexp "^From: \\(.*\\)") ; grab header field contents
  (save-excursion                     ; save state
    (let ((addr (car (rfc822-addresses  ; get address
                      (buffer-substring (match-beginning 1)
                                        (match-end 1)))))
          (buffer (get-buffer-create " *temp*")) ; set local variables
          folder)
      (set-buffer buffer)             ; jump to temporary buffer
      (unwind-protect                 ; run kill-buffer when done
          (progn                      ; function grouping construct
            (insert-file-contents (expand-file-name "aliases"
                                                    mh-user-path))
            (goto-char (point-min))   ; grab aliases file and go to start
            (setq folder
                  ;; Search for the given address, even commented-out
                  ;; addresses are found!
                  ;; The function search-forward-regexp sets values that are
                  ;; later used by match-beginning and match-end.
                  (if (search-forward-regexp (format "^;*\\(.*\\):.*%s"
                                                     addr) nil t)
                      ;; NOTE WELL: this is what the return value looks like.
                      ;; You can modify the format string to match your own
                      ;; Mail hierarchy.
                      (format "+%s" (buffer-substring (match-beginning 1)
                                                      (match-end 1))))))
        (kill-buffer buffer))          ; get rid of our temporary buffer
      folder)))                        ; function's return value

(setq mh-default-folder-for-message-function 'my-mh-folder-from-address)
```

The hook `mh-refile-msg-hook` is called after a message is marked to be refiled.

The variable `mh-sortm-args` holds extra arguments to pass on to the `sortm` command. Note: this variable is only consulted when a prefix argument is given to *M-x mh-sort-folder*. It is used to override any arguments given in a `sortm:` entry in your MH profile (`~/.mh_profile`).

3.4.3.1 Scan line formatting

The name of the program that generates a listing of one line per message is held in `mh-scan-prog` (default: ‘`"scan"`’). Unless this variable contains an absolute pathname, it is assumed to be in the `mh-progs` directory. You may link another program to `scan` (see `mh-profile`(5)) to produce a different type of listing.

If you change the format of the scan lines you'll need to tell mh-e how to parse the new format. As you see, quite a lot of variables are involved to do that. The first variable has to do with pruning out garbage.

`mh-valid-scan-line`
: This regular expression describes a valid scan line. This is used to eliminate error messages that are occasionally produced by `inc` or `scan` (default: ‘`"^ *[0-9]"`’).

Next, two variables control how the message numbers are parsed.

`mh-msg-number-regexp`
: This regular expression is used to extract the message number from a scan line. Note that the message number must be placed in quoted parentheses, (\\(...\\)), as in the default of ‘`"^ *\\([0-9]+\\)"`’.

`mh-msg-search-regexp`
: Given a message number (which is inserted in ‘`%d`’), this regular expression will match the scan line that it represents (default: ‘`"^[^0-9]*%d[^0-9]"`’).

Finally, there are a slew of variables that control how mh-e marks up the scan lines.

`mh-cmd-note`
: Number of characters to skip over before inserting notation (default: 4). Note how it relates to the following regular expressions.

`mh-deleted-msg-regexp`
: This regular expression describes deleted messages (default: ‘`"^....D"`’). See also `mh-note-deleted`.

`mh-refiled-msg-regexp`
: This regular expression describes refiled messages (default: ‘`"^....\\^"`’). See also `mh-note-refiled`.

`mh-cur-scan-msg-regexp`
: This regular expression matches the current message (default: ‘`"^....\\+"`’). See also `mh-note-cur`.

`mh-good-msg-regexp`
: This regular expression describes which messages should be shown when mh-e goes to the next or previous message. Normally, deleted or refiled messages are skipped over (default: ‘`"^....[^D^]"`’).

`mh-note-deleted`
: Messages that have been deleted to are marked by this string (default: ‘`"D"`’). See also `mh-deleted-msg-regexp`.

`mh-note-refiled`
: Messages that have been refiled are marked by this string (default: ‘`"^"`’). See also `mh-refiled-msg-regexp`.

`mh-note-copied`
: Messages that have been copied are marked by this string (default: ‘`"C"`’).

`mh-note-cur`
: The current message (in MH, not in mh-e) is marked by this string (default: ‘`"+"`’). See also `mh-cur-scan-msg-regexp`.

`mh-note-repl`
: Messages that have been replied to are marked by this string (default: ‘`"-"`’).

`mh-note-forw`
: Messages that have been forwarded are marked by this string (default: ‘`"F"`’).

`mh-note-dist`
: Messages that have been redistributed are marked by this string (default: ‘`"R"`’).

`mh-note-printed`
: Messages that have been printed are marked by this string (default: ‘`"P"`’).

`mh-note-seq`
: Messages in a sequence are marked by this string (default: ‘`"%"`’).

3.4.4 Printing Your Mail

Normally messages are printed in the foreground. If this is slow on your system, you may elect to set `mh-print-background` to non-`nil` to print in the background. If you do this, do not delete the message until it is printed or else the output may be truncated. The variable `mh-lpr-command-format` controls how the printing is actually done. The string can contain one escape, ‘`%s`’, which is filled with the name of the folder and the message number and is useful for print job names. As an example, the default is ‘`"lpr -J '%s'"`’.

3.4.5 Files and Pipes

The initial directory for the `mh-store-msg` command is held in `mh-store-default-directory`. Since I almost always run `mh-store-msg` on sources, I set it to my personal source directory like this:

```
(setq mh-store-default-directory (expand-file-name "~/src/"))
```

Subsequent incarnations of `mh-store-msg` offer the last directory used as the default. By the way, `mh-store-msg` calls the Emacs Lisp function `mh-store-buffer`. I mention this because you can use it directly if you're editing a buffer that contains a file that has been run through `uuencode` or `shar`. For example, you can extract the contents of the current buffer in your home directory by typing *`M-x mh-store-buffer`* `RET ~ RET`.

3.4.6 Finishing Up

The two variables `mh-before-quit-hook` and `mh-quit-hook` are called by *`q`* (`mh-quit`). The former one is called before the quit occurs, so you might use it to perform any mh-e operations; you could perform some query and abort the quit or call `mh-execute-commands`, for example. The latter is not run in an mh-e context, so you might use it to modify the window setup.

3.5 Searching Through Messages

If you find that you do the same thing over and over when editing the search template, you may wish to bind some shortcuts to keys. This can be done with the variable `mh-pick-mode-hook`, which is called when *`M-s`* (`mh-search-folder`) is run on a new pattern.

The string `mh-partial-folder-mode-line-annotation` is used to annotate the mode line when only a portion of the folder is shown. For example, this will be displayed after running *M-s* (`mh-search-folder`) to list messages based on some search criteria (see Section 2.5 [Searching], page 22). The default annotation of '`"select"`' yields a mode line that looks like:

```
--%%-{+inbox/select} 2 msgs (2-3)      (MH-Folder)--All-----------------
```

Appendix A Odds and Ends

This appendix covers a few topics that don't fit elsewhere. Here I tell you how to report bugs and how to get on the mh-e mailing list. I also point out some additional sources of information.

A.1 Bug Reports

The current maintainer of mh-e is Stephen Gildea <*gildea@lcs.mit.edu*>. Please mail bug reports directly to him, as well as any praise or suggestions. Please include the output of `M-x mh-version` (see Section 2.7 [Miscellaneous], page 25) in any bug report you send.

A.2 mh-e Mailing List

There is a mailing list, *mh-e@x.org*, for discussion of mh-e and announcements of new versions. Send a "subscribe" message to *mh-e-request@x.org* to be added. Do not report bugs on this list; mail them directly to the maintainer (see Section A.1 [Bug Reports], page 45).

A.3 MH FAQ

An FAQ appears monthly in the newsgroup '`comp.mail.mh`'. While very little is there that deals with mh-e specifically, there is an incredible wealth of material about MH itself which you will find useful. The subject of the FAQ is *MH Frequently Asked Questions (FAQ) with Answers*.

The FAQ can be also obtained by anonymous `ftp` or via the World Wide Web (WWW). It is located at:

```
ftp://rtfm.mit.edu/pub/usenet/news.answers/mail/mh-faq/part1
http://www.cis.ohio-state.edu/hypertext/faq/usenet/mail/mh-faq/part1/faq.html
```

Otherwise, you can use mail. Send mail to *mail-server@rtfm.mit.edu* containing the following:

```
send usenet/news.answers/mail/mh-faq/part1
```

A.4 Getting mh-e

If you're running a pre-4.0 version of mh-e, please consider upgrading. You can either have your system administrator upgrade your Emacs, or just the files for mh-e.

The MH distribution contains a copy of mh-e in `miscellany/mh-e`. Make sure it is at least Version 4.0.

The latest version of mh-e can be obtained via anonymous `ftp` from '`ftp.x.org`'. The file containing mh-e is currently `/misc/mh-e/mh-e-5.0.1.tar.Z`. I suggest that you extract the files from `mh-e-5.0.1.tar.Z` in the following fashion:

```
% cd                                  # Start in your home directory
% mkdir lib lib/emacs                 # Create directory for mh-e
% cd lib/emacs
% zcat path/to/mh-e-5.0.1.tar.Z | tar xvf -    # Extract files
```

To use these new files, add the following to `~/.emacs`:

```
(setq load-path (cons (expand-file-name "~/lib/emacs") load-path))
```

That's it! If you're already running Emacs, please quit that session and start again to load in the new mh-e. Check that you're running the new version with the command *`M-x mh-version`* after running any mh-e command. The distribution comes with a file called `MH-E-NEWS` so you can see what's new.

Appendix B History of mh-e

mh-e was originally written by Brian Reid in 1983 and has changed hands twice since then. Jim Larus wanted to do something similar for GNU Emacs, and ended up completely rewriting it that same year. In 1989, Stephen Gildea picked it up and is now currently improving and maintaining it.

B.1 From Brian Reid

One day in 1983 I got the flu and had to stay home from work for three days with nothing to do. I used that time to write MHE. The fundamental idea behind MHE was that it was a “puppeteer” driving the MH programs underneath it. MH had a model that the editor was supposed to run as a subprocess of the mailer, which seemed to me at the time to be the tail wagging the dog. So I turned it around and made the editor drive the MH programs. I made sure that the UCI people (who were maintaining MH at the time) took in my changes and made them stick.

Today, I still use my own version of MHE because I don’t at all like the way that GNU mh-e works and I’ve never gotten to be good enough at hacking Emacs Lisp to make GNU mh-e do what I want. The Gosling-emacs version of MHE and the GNU Emacs version of mh-e have almost nothing in common except similar names. They work differently, have different conceptual models, and have different key bindings.[1]

Brian Reid, June 1994

B.2 From Jim Larus

Brian Reid, while at CMU or shortly after going to Stanford wrote a mail reading program called MHE for Gosling Emacs. It had much the same structure as mh-e (i.e., invoked MH programs), though it was simpler and the commands were slightly different. Unfortunately, I no longer have a copy so the differences are lost in the mists of time.

In ’82-83, I was working at BBN and wrote a lot of mlisp code in Gosling Emacs to make it look more like Tennex Emacs. One of the packages that I picked up and improved was Reid’s mail system. In ’83, I went back to Berkeley. About that time, Stallman’s first version of GNU Emacs came out and people started to move to it from Gosling Emacs (as I recall, the transition took a year or two). I decided to port Reid’s MHE and used the mlisp to Emacs Lisp translator that came with GNU Emacs. It did a lousy job and the resulting code didn’t work, so I bit the bullet and rewrote the code by hand (it was a lot smaller and simpler then, so it took only a day or two).

Soon after that, mh-e became part of the standard Emacs distribution and suggestions kept dribbling in for improvements. mh-e soon reached sufficient functionality to keep me happy, but I kept on improving it because I was a graduate student with plenty of time on my hands and it was more fun than my dissertation. In retrospect, the one thing that I

[1] After reading this article, I questioned Brian about his version of MHE, and received some great ideas for improving mh-e such as a dired-like method of selecting folders; and removing the prompting when sending mail, filling in the blanks in the draft buffer instead. I passed them on to Stephen Gildea, the current maintainer, and he was excited about the ideas as well. Perhaps one day, mh-e will again resemble MHE, although none of these ideas are manifest in Version 5.0.

regret is not writing any documentation, which seriously limited the use and appeal of the package.

In '89, I came to Wisconsin as a professor and decided not to work on mh-e. It was stable, except for minor bugs, and had enough functionality, so I let it be for a few years. Stephen Gildea of BBN began to pester me about the bugs, but I ignored them. In 1990, he went off to the X Consortium, said good bye, and said that he would now be using `xmh`. A few months later, he came back and said that he couldn't stand `xmh` and could I put a few more bug fixes into mh-e. At that point, I had no interest in fixing mh-e, so I gave the responsibility of maintenance to him and he has done a fine job since then.

Jim Larus, June 1994

B.3 From Stephen Gildea

In 1987 I went to work for Bolt Beranek and Newman, as Jim had before me. In my previous job, I had been using RMAIL, but as my folders tend to run large, I was frustrated with the speed of RMAIL. However, I stuck with it because I wanted the GNU Emacs interface. I am very familiar and comfortable with the Emacs interface (with just a few modifications of my own) and dislike having to use applications with embedded editors; they never live up to Emacs.

MH is the mail reader of choice at BBN, so I converted to it. Since I didn't want to give up using an Emacs interface, I started using mh-e. As is my wont, I started hacking on it almost immediately. I first used version 3.4m. One of the first features I added was to treat the folder buffer as a file-visiting buffer: you could lock it, save it, and be warned of unsaved changes when killing it. I also worked to bring its functionality a little closer to RMAIL. Jim Larus was very cooperative about merging in my changes, and my efforts first appeared in version 3.6, distributed with Emacs 18.52 in 1988. Next I decided mh-e was too slow and optimized it a lot. Version, 3.7, distributed with Emacs 18.56 in 1990, was noticeably faster.

When I moved to the X Consortium I became the first person there to not use xmh. (There is now one other engineer there using mh-e.) About this point I took over maintenance of mh-e from Jim and was finally able to add some features Jim hadn't accepted, such as the backward searching undo. My first release was 3.8 (Emacs 18.58) in 1992.

Now, in 1994, we see a flurry of releases, with both 4.0 and 5.0. Version 4.0 added many new features, including background folder collection and support for composing MIME messages. (Reading MIME messages remains to be done, alas.) While writing this book, Bill Wohler gave mh-e its closest examination ever, uncovering bugs and inconsistencies that required a new major version to fix, and so version 5 was released.

Stephen Gildea, June 1994

Appendix C Changes to mh-e

mh-e had a fairly major facelift between Versions 3 and 4. The differences between Versions 4 and 5 from the user's viewpoint are relatively minor. The prompting order for the folder and message number in a couple of functions had been switched inadvertently in Version 4. Version 5 switches the order back. The `+inbox` folder is no longer hard-coded, but rather uses the '`Inbox`' MH Profile entry. See the file `etc/MH-E-NEWS` in the Emacs distribution for more details on the changes.

This section documents the changes between Version 3 and newer versions so that you'll know which commands to use (or which commands you won't have) in case you're stuck with an old version.

The following tables summarize the changes to buffer names, commands and variables.

Buffer Mode Names

Version 3	**Version 4**
`mh-e folder`	`MH-Folder`
`mh-e scan`	`MH-Folder`
`mh-e show`	`MH-Folder Show`
`Fundamental`	`MH-Show`
`mh-e letter`	`MH-Letter`
`mh-e letter`	`MH-Pick`

Commands

Version 3		Version 4	
Function	**Command**	**Command**	**Function**
`mh-first-msg`	`<`	`M-<`	`mh-first-msg`
`-`	`-`	`M->`	`mh-last-msg`
`mh-show`	`.`	`RET`	`mh-show`
`-`	`-`	`,`	`mh-header-display`
`mh-reply`	`a`	`r`	`mh-reply`
`mh-redistribute`	`r`	`M-d`	`mh-redistribute`
`mh-unshar-msg`	`-`	`M-n`	`mh-store-msg`
`mh-write-msg-to-file`	`M-o`	`C-o`	`mh-write-msg-to-file`
`mh-delete-msg-from-seq`	`C-u M-%`	`M-#`	`mh-delete-seq`
`-`	`-`	`M-q`	`mh-list-sequences`
`mh-quit`	`b`	`q`	`mh-quit`
`-`	`-`	`C-C C-f C-r`	`mh-to-field ('From:')`
`-`	`-`	`C-C C-f C-d`	`mh-to-field ('Dcc:')`

Variables

Version 3		Version 4	
Variable	**Value**	**Value**	**Variable**
`mh-show-buffer-mode-line-buffer-id`	`"{%%b}  %s/%d"`	`"{show-%s} %d"`	`mh-show-buffer-mode-line-buffer-id`
`mh-unshar-default-directory`	`""`	`nil`	`mh-store-default-directory`

New Variables

`mail-citation-hook`
`mail-header-separator`
`mh-auto-folder-collect`
`mh-comp-formfile`
`mh-repl-formfile`
`mh-delete-msg-hook`
`mh-forward-subject-format`
`mh-inc-prog`
`mh-mime-content-types`
`mh-default-folder-for-message-function`
`mh-mhn-args`
`mh-new-draft-cleaned-headers`
`mh-pick-mode-hook`
`mh-refile-msg-hook`
`mh-scan-prog`
`mh-send-prog`
`mh-show-hook`
`mh-show-mode-hook`
`mh-signature-file-name`
`mh-sortm-args`
`mh-repl-formfile`

Appendix D GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

D.1 Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

D.2 TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

D.3 How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and an idea of what it does.
Copyright (C) 19yy  name of author

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) 19yy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details
type `show w'.  This is free software, and you are welcome
to redistribute it under certain conditions; type `show c'
for details.
```

The hypothetical commands '`show w`' and '`show c`' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than '`show w`' and '`show c`'; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright
interest in the program `Gnomovision'
(which makes passes at compilers) written
by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Function Index

Variable Index

Concept Index

Table of Contents