The Elm Users Guide

A painless introduction to electronic mail
using the **Elm** *mail system*

The Elm Mail System
(Version 2.4)

Syd Weinstein, Elm Coordinator
Datacomp Systems, Inc.
3837 Byron Road
Huntingdon Valley, PA 19006-2320

email: elm@DSI.COM or dsinc!elm

ABSTRACT

While various UNIX utilities have been designed to take advantage of the CRT screen (rather than line-oriented systems) electronic mail systems haven't ‘‘caught the wave’’. This document presents Elm, a sophisticated full-screen mail system for interactive use. Elm offers all the features of */bin/mail*, *Berkeley Mail*, *uumail*, *NMail* and the AT&T Mail family of UNIX mailers (namely *PMX/TERM*) in a unified and intuitive fashion.

Elm Users Guide

(The Elm Mail System, Version 2.4)

October 1, 1992

Syd Weinstein
Datacomp Systems, Inc.
3837 Byron Road
Huntingdon Valley, PA 19006-2320

email: elm@DSI.COM or dsinc!elm

Derived from
''The Elm Mail System, Version 2.0''
by
Dave Taylor
Intuitive Systems
Mountain View, California
email: taylor@intuitive.com or limbo!taylor

1. Introduction

The UNIX® operating system was originally written on a small PDP machine, to work with teletypewriters. Consequently, all the original software for the operating system was written with a ''tty'' in mind; with line oriented interfaces, rather than screen oriented.

Gradually this began to change, and, with the addition of the Berkeley *vi* visual shell to the *ed* line editor, the trend began to be brought to the attention of UNIX systems designers. And yet, this transition has seemed too subtle for many software engineers, and so precious few UNIX programs are available designed to exploit the entire CRT screen.

Consequently, after becoming more and more disenchanted with the existing electronic mail facilities (*binmail* and *Berkeley Mail*) Dave Taylor decided to undertake creating his own system. Once the ball was rolling it became quite difficult not to keep enhancing it, especially as he and the Elm Development Group became more and more familiar with various different mail systems, but the result, we hope, is a solid, unified interface to electronic mail suitable for almost any environment.

®UNIX is a registered trademark of UNIX Systems Laboratories.

One thing that sets the Elm mail system off from other packages is that it comes with an extensive documentation set, including; *The Elm Users Guide*, which you're holding, *The Elm Reference Guide*, *The Elm Alias Users Guide*, *The Elm Filter System Users Guide*, *The Elm Forms Mode Guide*, and *The Elm Configuration Guide*. The names of each indicate what they discuss (not unreasonably).

This particular document contains; this introduction, a tutorial session with the mailer, a discussion of non-interactive uses of Elm, a brief foray into the Elm utilities, and finally, credits and references.

The Elm Reference Guide contains a much more in-depth discussion of all the possible options available within the Elm environment, including the *.elm/elmrc* file, the available outbound mail processing features, a section for expert users, suggestions on debugging strange installations, and many other useful topics.

The Elm Alias Users Guide is the place to go to learn about the format of the *.elm/aliases.text* file and the associated data files, system aliases, and other related topics.

One of the more innovative features of Elm is the ability to transmit and receive ‘‘forms’’ (as defined by the AT&T Mail system). To learn about how to create and reply to forms you should refer to *The Elm Forms Mode Guide*.

If you find yourself getting too much mail delivered in your mailbox, you might be a good candidate for the *filter* program. To learn more about what this program is and how to use it please consult *The Elm Filter System Users Guide*.

Finally, if you're working with the actual source code and are interested in what all the locally configured options are and mean, please see *The Elm Configuration Guide*. It is strongly suggested that all system administrators and people installing the system print out the local *sysdefs.h* file and have it handy while reading the configuration guide.

The recommended order of learning the Elm system is to skim this guide until you feel confident enough to start up the program. Once that's reached, *use it!* Soon you'll find areas that you need to expand into, and you can achieve that by going into the options screen within Elm and changing your user level accordingly. After a while, sit down again and read through this guide. You should have a couple of ‘‘aha!’’ intuitive flashes. After another period of time, you're ready to delve into the real power of the system and read the reference guide. The other manuals should be read as the need arises.

In any case, the system should be usable without reading *any* of the documentation!

Any comments or problems with any of the documentation or the program itself are welcome; if you can't get electronic mail to the Elm Development Group, please feel free instead to drop me a note via the overland mail address in the title page.

2. The Main System

Upon entry to the program, the main screen will be displayed as below:

```
            Mailbox is ´/usr/mail/mymail´ with 15 messages  [Elm 2.4PL22]
->   N     1  Apr 24  Larry Fenske   (49)    Hello there
     N     2  Apr 24  jad@hpcnoe     (84)    Chico?  Why go there?
     E     3  Apr 23  Carl Smith     (53)    Dinner tonight?
     NU    4  Apr 18  Don Knuth      (354)   Your version of TeX...
     N     5  Apr 18  games          (26)    Bug in cribbage game
      A    6  Apr 15  kevin          (27)    More software requests
           7  Apr 13  John Jacobs    (194)   How can you hate RUSH?
      U    8  Apr 8   decvax!mouse   (68)    Re: your Usenet article
           9  Apr 6   root           (7)
     O    10  Apr 5   root           (13)

      You can use any of the following commands by pressing the first character;
      d)elete or u)ndelete mail, m)ail a message, r)eply or f)orward mail, q)uit
       To read a message, press <return>.  j = move down, k = move up, ? = help
Command : @
```

There are a number of things to notice about this, the main screen;

- Most likely, on your computer the message currently ‘‘active’’ will be displayed in an inverse bar rather than being delimited by the two character arrow as here. It’s nothing to worry about; displaying inverse video is just quite difficult on printable guides!

- The first line on the screen always displays the name of the current folder, the number of messages in the folder, and the current Elm version number.

- The arrow (‘‘–>’’) or inverse video bar will always indicate which is the current message.

- The first field that appears associated with each message is the status field. This can be blank (as with most of the ones above, or can have any combination of the following:

  The first character signifies temporary status:

E	for an *expired* message
N	for a *new* message
O	for an *old* (i.e. not new but not read) message
D	for a *deleted* message

  The second character which signifies permanent status:

C	for *confidential* mail
U	for *urgent* mail
P	for a *private* message
A	for messages that have an *action* associated with them
F	for a *form* letter
M	for a *MIME* compliant message

The third character of the status field can be a '+' to indicate that the message is *tagged* too.

- Continuing from left to right, the next field is the message number. For the most part you can ignore these unless you want to quickly move to a specific message (as we'll see later).

- The date associated with each message is typically the date the person actually *sent* the message.

- The next field displayed indicates whom the message is from. Elm will try to display the *full name* of the person who sent the message, rather than the their return address or computer login. Some systems don't generate the correct headers, though, hence messages like numbers 2 and 8, where it's their return address.

- The number in parentheses is the total number of lines in the message.

- The final field is the subject of the message. Notice that messages might not have any subject, as in messages #9 and #10.

- A maximum of ten messages are displayed at one time.[1] Further into the document we'll learn how to change ''pages'' in the folder.

- The three line menu display will always indicate the relevant commands. There are actually two possible menus that can be displayed, based on the *user level* as set from either the options screen or the *.elm/elmrc* file. The alternate menu, for more advanced users, lists more options;

```
        |=pipe, !=shell, ?=help, <n>=set current to n, /=search pattern
  a)lias, C)copy, c)hange folder, d)elete, e)dit, f)orward, g)roup reply, m)ail,
    n)ext, o)ptions, p)rint, r)eply, s)ave, t)ag, q)uit, u)ndelete, or e(x)it
```

- Finally, the '@' character indicates where the cursor would be, awaiting your input.

The typical action at this point is to use the *<return>* key to read the current message, which will clear the screen and display the current message:

```
Message 1/15 from Larry Fenske                              Apr 24 ´87 at 8:45 pm edt
                                     Hello there
Dave,
     Just wanted to drop you a brief note to see what was going on with
you this afternoon.  Life here has been the usual fun and games...
        Ah well, off to the great wilds beyond the desk!
   Larry
Command (´i´ to return to index): @
```

Before we go further with our example, however, let's very quickly look at all the functions available from the main screen:

1. On screens with more than 24 lines, additional messages are displayed automatically. Furthermore, if you choose to turn off the menu display, you can have an even greater number displayed.

Command	**Action**
<return> or <space>	Read current message.
|	Pipe current message or tagged messages to specified system command.
!	Shell escape.
$	Resynchronize folder.
?	Help mode — any key pressed will be explained.
+ or <right>	Display next page of subjects.
– or <left>	Display previous page of subjects.
=	Set current message to 1.
*	Set current to last message.
<number><return>	Set current message to number *number*.
/	Search for pattern in subject/from lines.
//	Search for pattern in entire folder.
<	Scan message for calendar entries.[2]
>	A synonym for **s — save** message or messages.
a	Alias, change to ‘‘alias’’ mode.
b	Bounce — remail message (see **f — forward** too).
C	Copy current message or tagged messages to folder.
c	Change to another folder.
d	Delete current message.
<control>-D	Delete all messages matching specified pattern.
e	Edit current folder, resyncing upon re-entry.[3]
f	Forward message to specified user.[4]
g	Group reply — reply to everyone who received the current message.
h	Display message with headers.
i	Return to index screen after displaying message.
J	Set current to next message.
j or <down>	Set current to next message not marked deleted.
K	Set current to previous message.
k or <up>	Set current to previous message not marked deleted.
l	Limit displayed messages based on the specified criteria.
<control>-L	Rewrite screen.
m	Mail to arbitrary user(s).

2. Some sites might opt not to have the calendar feature available.
3. Some sites might opt not to have the edit folder feature available.
4. The difference between **forward** and **bounce** is rather subtle — a forwarded message is *from* you, with the original message included, whereas a bounced message is still from the original sender.

n	Read current message, then increment to next message not marked deleted.
o	Alter current system options.
p	Print current message or tagged messages.
q	Quit — maybe prompting for messages to delete, store, or keep.
Q	Quick quit — like quit but without prompting.
r	Reply to the author of current message.
s	Save current message or tagged messages to folder.
t	Tag current message.
<control>-T	Tag all messages matching specified pattern.
u	Undelete current message.
<control>-U	Undelete all messages matching specified pattern.
x	Exit — prompt if mailbox changed, don't record as read, don't save.
X	Exit immediately — don't record as read, don't save.

But let's go back to our example and see some of this at work, shall we?

We were reading the message from Larry and the screen looked like:

```
Message 1/15 from Larry Fenske                              Apr 24 ´87 at 8:45 pm edt
                                         Hello there
Dave,
     Just wanted to drop you a brief note to see what was going on with
you this afternoon.  Life here has been the usual fun and games...
                      Ah well, off to the great wilds beyond the desk!
                Larry
Command (´i´ to return to index): @
```

From this point let's go ahead and reply to the message. To do this, we can use the **reply** command. To do this, we type 'r' and the last few lines change to something like:

```
--------------------------------------------------------------------------------------------------------------
Command: Reply to message                                        Copy message? (y/n) @
```

To which we decide that we don't need the text of this message in our new one, so we reply *no* by pressing the 'n' key. The bottom part of the window then changes to:

```
--------------------------------------------------------------------------------------------------------------
Command: Reply to message          To: Larry Fenske
Subject: Re: Hello there@
```

At this point we can either decide to enter a new subject (you could use either *backup word* (*<control>-W*) or *backup line* (this is the same as your ''line kill'' character, *<control>-X* or *<control>-U* usually)) or go with this one by pressing *<return>*. We'll just leave it as it is and press *<return>*, changing the screen to:

```
--------------------------------------------------------------------------------------------------------------
Command: reply to message          To: Larry Fenske
Subject: Re: Hello there
Copies to: @
```

There's no one we want to have receive copies of this message, so we'll just press <*return*> again to indicate this.

Once you've answered these questions the program will put you into your favorite editor and let you compose a response. When you're done it then asks:

```
Please choose one of the following options by parenthesized letter: @
                e)dit message, edit h)eaders, s)end it, or f)orget it.
```

Since we're just interested in sending the message out, we'll choose the **send** option and press 's'. The program then sends the message, indicating that by the line below saying:

```
    Sending mail...
```

then putting at the bottom of the screen ''Mail sent!'' and giving us the prompt:

```
----------------------------------------------------------------------------------------------------
Command:                                                          (Use ´i´ to return to index.)
```

Pretty easy, isn't it? Let's continue by going back to the main screen pressing the 'i' key to request the *index*. The screen is then:

```
             Mailbox is ´/usr/mail/mymail´ with 15 messages  [Elm 2.4PL22]
->        1  Apr 24  Larry Fenske  (49)   Hello there
     N    2  Apr 24  jad@hpcnoe    (84)   Chico?  Why go there?
     E    3  Apr 23  Carl Smith    (53)   Dinner tonight?
     NU   4  Apr 18  Don Knuth     (354)  Your version of TeX...
     N    5  Apr 18  games         (26)   Bug in cribbage game
      A   6  Apr 15  kevin         (27)   More software requests
          7  Apr 13  John Jacobs   (194)  How can you hate RUSH?
      U   8  Apr 8   decvax!mouse  (68)   Re: your Usenet article
          9  Apr 6   root          (7)
     O   10  Apr 5   root          (13)

      You can use any of the following commands by pressing the first character;
      d)elete or u)ndelete mail, m)ail a message, r)eply or f)orward mail, q)uit
       To read a message, press <return>.  j = move down, k = move up, ? = help
Command : @
```

Notice that the first message is no longer marked as *new* since we've now read it.

Let's go ahead and read the message from kevin (message #6) since it has some sort of *action* associated with it anyway. To do this, we simply press the '6' key, which will change the bottom of the screen to:

```
Command: New Current Message                               Set current message to : 6@
```

We'll just press <*return*> to move the active message pointer (the arrow).

Now we're pointing at the new message, so let's go ahead and read it by pressing <*return*> again, giving us:

```
Message 6/15 from kevin                                   Apr 15 ´87 at 11:36 am pst
                               More software requests
Action: please acknowledge receipt
I don´t suppose you have a nifty netnews reader around too, do you??
--
kevin
Command (´i´ to return to index): @
```

Well, it turns out that we don't, but a friend of ours does, so let's **forward** the message to them by pressing 'f':

```
Command: Forward message                               Edit outgoing message (y/n) ? @
```

No need to edit it, so let's answer *no* with 'n':

```
Command: Forward message                              Edit outgoing message (y/n) ? No
To: @
```

We type in the address of the person we're forwarding to (in this case `usenet`) and press <*return*>:

```
Command: Forward message              To: usenet
Subject: More software requests (fwd)@
```

To which we again press <*return*> to take the default subject. Elm asks for any possible copy recipients, asks us if we're sure we want to send it, and shoots it off.

Enough mail for now, however, so we just **exit** by pressing the 'x' key at the ''Command:'' prompt and the program drops us back into our shell.

As you can see, it's quite easy to use the Elm system, so rather than continue with our example, let's look at some other aspects of the program.

3. Non-Interactive Uses of Elm

Before we go ahead and discuss the Elm utilities, it's worth noting that there are a couple of other ways to use the main mail system, namely to ''send only'' and to send files (batchmail).

To send a message to someone without any of the associated overhead of reading in a mail folder, you can invoke the mailer with the name(s) of the people to send to. For example:

$ **elm dave_taylor**

Elm then prompts for Subject, Copies, and then drops you in your editor (defined in the *.elm/elmrc* file) to compose the message. When composition is complete, the program verifies transmission then terminates.

Elm also supports batch type mailing, of files and so on, by using the following command:

$ **elm dave_taylor < help.c**

which reads in the file and transmits it to the specified user.

A subject may be specified with ‘‘–s *subject*’’ as an option to Elm in either ‘‘send only’’ or ‘‘batch’’ modes, as in:

$ **elm –s "File help.c, as promised" dave_taylor < help.c**

Elm also has an option to specify that a certain file be used as the initial text of the message. This makes it easier to use Elm with other programs that interface with a mailer. Use ‘‘–i *file*’’ as an option to Elm in ‘‘send only’’ mode. For example, to specify to *rn* to use Elm as the mailer, define the following in **RNINIT**:

```
–EMAILPOSTER="elm –i %h –s \"Re: %S\" %t"
```

See the man page for *rn*(1) or the newsreader you use for more specific information.

4. The Elm Utilities

The Elm mail system is really much more than a single program for reading and writing mail. It’s a unified environment for intelligently dealing with electronic correspondence. As a consequence, it contains a number of other programs to aid in the easy processing of ‘‘email’’, including the following;

answer — An answering-machine transcription program. Please see the manual entry for more information on how to use this program.

autoreply — This program, in conjunction with the *arepdaemon*, enables users to automatically reply to incoming mail when they’re not directly available.

checkalias — A script for checking aliases simply.

elmalias — A tool to obtain information on defined aliases.

listalias — A script to produce sorted listings of aliases.

messages — A script that lists the number of messages in the specified folder. Suitable for login scripts and such.

newmail — This program monitors a mailbox or set of mailboxes and can output notification of new mail in one of two possible formats; either:

```
>> New mail from Jimmy Carter –– Urgent matters of state
>> New mail from joe@ucbvax.arpa –– Did I hear someone say PARTY??
```

if running as *newmail* or:

```
Jimmy Carter –– Urgent matters of state
joe@ucbvax.arpa –– Did I hear someone say PARTY??
```

if running as *wnewmail*. BSD Users will find this is a far superior *biff* program.

wnewmail — This is the same program as *newmail*, but has different defaults if invoked this way.

readmsg This handy little program can be used in two ways. First off, it can be used to easily read a specific message in the incoming mailbox, specified by ordinal number, by a string that the message contains, by the metacharacter '$' which represents the last message in the folder, or '*' which matches all the messages in the folder.

For example;

$ **readmsg 4 5 9 | lpr**

would generate a printout, *sans* superfluous headers, of the fourth, fifth and ninth messages in your mailbox.

$ **readmsg Carter | page**

would be a fast way to read what 'ole Jimmy has to say, and

$ **readmsg –h hear someone say**

would list, including all the message headers, the message containing the string ''hear someone say''.

frm Similar to the Berkeley *from* command, this will give you a ''table of contents'' or a summary of either the current mailbox or a mailbox of your choice. It's useful to see what mail is pending, or what's in a mailbox. If used with the ''–n'' option, it will number each message in a way compatible with the arguments *readmsg* expects to get, too!

newalias This is used to install new user/group aliases. Please see *The Elm Alias Users Guide* for further information.

fastmail Sometimes you want to have a batchmailing system that works as quickly as possible, not bothering with aliases or any of the other ''deluxe'' features of Elm. An example of this would be if you have a large mailing list for a newsletter, say. This command, *fastmail*, is designed just for that purpose. It avoids the lengthy startup time of Elm while still generating valid RFC-822[5] mail. Please see the manual entry for more information on this command.

5. Credits and So on — Elm's early years from Dave Taylor

This mailer has been evolving over the past few years with invaluable comments on the interface and general mail issues from the following people; Jim Davis, Steve Wolf (or should that say Steve!! Wolf!!!! perhaps?), Larry Fenske, Rob Sartin, John Dilley and Carl Dierschow.

For the Berkeley implementation, lots of help came from both John Lebovitz and Ken Stone.

For the Amdahl/UTS implementation, thanks to Scott McGregor and Stan Isaacs.

For the Sun problems, Dave England in Lancaster (UK) turned out to be ''bloody useful'', as he would no doubt say.

The Pyramid version is due to the work of Steve Malone of the University of Washington.

5. If you don't know what RFC-822 is, don't worry about it!

A number of other people have been amazingly disciplined at reporting problems and (usually, much to my relief) indicating the fixes needed, especially Guy Hillyer, Bruce Townsend and Eric Negaard.

There have been many, many others, too numerous to mention, that have inspired me with questions like ‘‘Why can’t Elm...’’ or ‘‘Why does it ...’’ or ‘‘Can we make it...’’ too. A loud round of applause and an enthusiastic THANK YOU to you all!!

Also helpful was the ability to have my ‘‘own’’ machine to close up the many many iterative loops that this software has gone through — since being on a big multi-user machine tends to slow it down to a standstill. For that, I thank Hewlett-Packard Colorado Networks Division for their initial support, and now HP Laboratories for the wonderful working environment that I have here (more than any one person is safe having, I suspect).

Mailers that have influenced the design and implementation of Elm, for better or worse, include the following;

/bin/mail The most basic of mailers, this one was simply the example of how *not* to force the user to interact with a program.

Berkeley Mail A surprisingly sophisticated mailer, especially the version with 4.3 BSD, *Berkeley Mail* still suffers from the tendency to force the user to know too much about the system, and also lacks a friendly, non-cryptic interface for the large subset of the population that use UNIX but aren’t interested in becoming a ‘‘hacker’’.

MH This is another nifty mailer. The main difference between this and the other mailers about is that it is a discrete set of programs rather than a single unified interface. This is quite useful for those people that receive *lots* of mail and are willing to spend some time learning the intricacies of the program. It’s quite powerful, but again, misses some of the basic friendly features the majority of users are interested in.

dmail A sort of mutated cross between *MH* and *Berkeley Mail*, it was this program that convinced me to implement the **limit** functions.

NMail A part of the HP AI Workstation Software Environment, this program hints at the power that mailers could be given to help deal with mail in a quick and intelligent fashion. Most of what it can do, and a lot more, are therefore part of Elm. Indubitably. And quite fast too!

PMX/TERM Part of the AT&T Mail package, a single glance at this interface convinced me that a PC interface, with almost half of the screen taken up by a silly function key mapping, is most certainly *not* the way to do things!! On the other hand, it also shows that the ‘‘forms’’ mode can be quite nicely integrated into a more sophisticated mailer.

fmail A nickel addition to the */bin/mail* program to add some rudimentary screen interface stuff, this nonetheless interesting mailer is part of the AT&T Toolchest.

Laurel A program only available within Xerox PARC, this was an interesting early attempt at a graphics based full-screen mail program. The one, perhaps trivial, part of the interface I really like was the fact that it would *cross out* a message that was marked for deletion. One day when we get real graphics and the ability to use them in a facility (not too) like *termcap*, perhaps Elm will do this too!

Mailtool Part of the Sun distribution package, this program is a really nice example of what can be done by putting a smart shell on top of a dumb program — it uses */bin/mail* for all the ‘‘dirty work’’.

Other than that, it's not a particularly interesting interface or mailer, and it certainly doesn't add much functionality!

Metamail What can I say? This isn't even a real mailer,[6] but is just what I dream of as an interface to mail in the future. A program that works sort of like HAL did in *2001: A Space Odyssey* — where it interrupts me as appropriate for important mail, and can answer inane and other queries itself according to what I teach it. Maybe Elm, by some sort of groupmind metamorphosis, will change into that someday. Maybe not. Maybe no one bothers to read this far into the document!!

Finally, it's also been a very useful experience overall, confirming my beliefs that iterative software design and keeping a close watch on users (not to mention an open mind and mailbox!) improves a product manyfold. Comments, feedback and bug reports (and fixes!) are, of course, always welcome!

6. Credits and So on Part II — The Usenet Elm Development Group

As coordinator of the Elm Development Group, I must add to the credits. This group of volunteers has taken the Elm code from Dave Taylor and added features, made it more robust, and more portable.

For getting the Usenet Elm group going, for without which I wouldn't be writing this, thanks to Greg Hackney at Southwestern Bell Telephone Co.

For the Configuration system and its newer Dist 2.0 version, thanks are due to Larry Wall of JPL-NASA.

For his over two hundred sets of changes to version 2.2, Elm owes a lot to Rob Bernardo who was at Pacbell at that time.

The ability to run Elm at all on Intel 286 machines owes a lot to the insistent nudging of the coordinator by Chip Salzenberg, Chip Rosenthal, and Tim Evans.

For getting us the rights to use the news macros and converting all the documentation over to those macros, thanks to Mike Brown.

For pushing us into the '90s and forcing the issues of NLS and foreign language support, I thank Jan Saell of Administration & Systemkonsult AB along with Larry Philps of SCO Canada. I'd also like to thank the many current and prior members of the Elm development and testing groups. The list varies from time to time as people come and go. Also, to all the Elm users out there who send in patches. The group can use all the extra help it gets. As a postScript, if you wish to join the group, all it takes is some time to work on things, and to ask.

Syd Weinstein, Elm Coordinator.

6. Not to be confused with the multi-media mailer, *metamail*, from Nathaniel Borenstein of Bellcore.