The Elm Alias System Users Guide

What aliases are and how to use them
in 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

The Elm Alias System 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

This document is intended as a supplement to the *Elm Users Guide* and is only of interest to those users desiring more knowledge about how aliases work and how to create strange and exciting aliases for their systems (okay, so maybe it's not *that* exciting!)

This document is broken up into the following sections; user aliases, group aliases, system aliases, editing and installing new aliases, general warnings and other chit-chat.

1. User Aliases

The simplest type of aliases in the Elm system are individual user aliases. These are made up of three parts;

aliasname list = username = address

Where the *aliasname list* is either a single aliasname[1] or a list of aliasnames separated by commas. Aliasnames are case insensitive and will be converted to all lower case by the alias system.

---

[1] Aliasnames can be any combination of letters, numbers, hyphens (‘–’), periods (‘.’), or underscores (‘_’). Letter case is not significant, that is, ‘‘FRED’’, ‘‘Fred’’, and ‘‘fred’’ are identical.

Username is used to indicate the full ‘‘real name’’ of the user. For example, if you had an alias for ‘‘taylor’’ to get to me, the *username* field would contain ‘‘Dave Taylor’’ or perhaps ‘‘Dave Taylor at HP’’ or some other permutation. The name can optionally be entered in the format of ‘‘Taylor; Dave’’. When this is done, the alias system can sort aliases on a last name basis for presentation on the alias menu within Elm. Elm uses this information to add the user’s full name to the addresses of outbound mail to get more readable addresses. No matter which format a name is entered, Elm displays it and puts it in mail headers the way you would expect (i.e., ‘‘Dave Taylor’’).

Address is either the user’s full electronic mail address or, if your system uses a smart mailer, the minimum address needed to specify the destination. For example, say my smart mailer contained information on how to get to machine ‘‘hp-sdd’’ and I wanted to have an address for my friend Ken there — I could have his address specified as simply ‘‘ken@hp-sdd’’ (or alternatively ‘‘hp-sdd!ken’’ since the two are functionally equivalent).

Let’s get on to some examples, shall we?

Consider this excerpt from my own *.elm/aliases.text* file;

```
# Mail aliases for Dave Taylor
# Friends from UCSD
addie          = Addie Silva    = addie@hp-sdd.SDD.HP.COM
frank,minjarez = Frank Minjarez = Minjarez.Verify@dockmaster.ddn.mil
pizzini        = Ken Pizzini    = hplabs!ucbvax!ucdavis!pai!ken
george         = Burdell; George P., Famous GT Guy= gburdell@matd.gatech.edu
```

Note that the alias for Frank Minjarez has two *aliasnames* associated with it, *frank* and *minjarez*. Also notice that the first and second aliases use the Internet style naming convention of *user@machine* whereas the third uses the *uucp* notation of *machine!user*. The fourth alias illustrates two features for the *username* field. An individual’s name can be entered in the form of *lastname; firstname, comment* which allows the alias system to be able to sort alias names on a last name basis. Additionally, any text after the comma is treated as a comment. It is stored in the alias file and displayed on the alias menu but will not appear in the comment field (between the parentheses) in the header of a message. This allows you to enter useful notes about an individual that you would not otherwise want to include in the **To:** header of a mail message.

For the most part, the notational format is transparent within the alias file — the only time it *does* make a difference is if you have to specify more than the machine that the user is receiving mail on (as in the third example above).

Suffice to say that if you must specify any sort of uucp route that you should use the uucp notation as much as possible to ensure that the system expands the correct machine name. Similarly, if you’re bouncing mail through different Internet sites (to get to a foreign system, as in the example below) you should use the notation that system expects, e.g.:

```
listserv%byuadmin.BITNET@rugters.edu
```

2. Group Aliases

After the confusion of user aliases, group aliases are even more fun! For the most part the notation is very similar:

aliasname list = *groupname* = *address list*

Where *aliasname list* and *groupname* are exactly equivalent to the corresponding fields in user aliases.

The interesting part is the *address list* field; this field is actually in the same notation as the aliasname list (i.e., a list of addresses separated by commas), so it’s really quite easy to create. It’s best to illustrate by example:

```
friends, mypals = The Gang of Six = joe, larry, mary, joanna,
                                    nancy, michael
```

Notice that you can continue onto as many lines as you'd like so long as each additional line starts with either a <*space*> or a <*tab*> character.

At times it may be desirable to send a message to everyone in the group except one or two people. This can be accomplished by adding the individuals' actual IDs to the **To:** list, preceded by a minus sign ('–'). For example, if you were having a surprise party for *nancy* from the above group and you wanted to discuss the details with the others, you could mail the message to:

```
friends –nancy
```

Unlike in prior versions of Elm, group alias addresses need not be *previously defined aliases* or valid mail addresses on the current machine. Any valid delivery address, local user, or prior user or group alias may now be part of a group alias *address list*.

3. System Aliases

System aliases are functionally equivalent to the individual Elm alias lists each Elm user has (both user aliases and group aliases) but are *read only* for everyone but the Elm administrator. The format of the file is identical to the users' files, and the only difference is that this file is expected to be located in the directory that contains the **system_hash_file** and **system_data_file** files (see the *Elm Configuration Guide* for more details on these variables).

Simply create the system alias file in the specified directory as you would a normal alias file, and install it the same way (see the following section for more details on that). Voila!!

4. Editing and Installing New Aliases

To install new aliases, you need merely to create, or modify, the file *$HOME/.elm/aliases.text* until you're satisfied with it and it meets the requirements discussed above. You can then try to install it with the command:

```
$ newalias
```

which either reports the number of aliases installed or the errors encountered trying to parse and store the given alias list.

Note that blank lines are no problem and that comments are not only allowed but actually encouraged, and must have '#' as the first character of each comment line.

5. Other Stuff not Covered Yet

Probably the biggest question you have in your mind right now is ''But how the heck does this relate to my existing *Berkeley Mail* aliases and the lower-level *sendmail* alias system?'' Well, rest assured that if you *really* want to have your aliases down in the transport you can. No problem. All you'll need to do is to turn off the address validation in Elm.

And for those ex-*Berkeley Mail* fans, you can translate your aliases into the format that Elm wants by running them through the *awk* script listed in the appendix.

Finally, if you have any problems or questions, try looking in the *newalias* manual entry.

Appendix One
An *awk* Script for Translating Aliases from a Berkeley Mail *.mailrc* File to an Elm *.elm/aliases.text* File

```
BEGIN { print "# Elm .elm/aliases.text file, from a .mailrc file..."
      print ""
      }
next_line == 1 {
      next_line = 0;
      group = ""
      for (i = 1; i <= NF; i++) {
        if (i == NF && $i == "\\") sep = ""
        else                       sep = ", "
        if ($i == "\\") {
          group = sprintf("%s,", group)
          next_line = 1;
        }
        else if (length(group) > 0)
          group = sprintf("%s%s%s", group, sep, $i);
        else
          group = $i;
        }
        print "\t" group
      }
$1 ˜ /[Aa]lias | [Gg]roup/ {
      if ( NF == 3)
        print $2 " = user alias = " $3;
      else {
        group = ""
        for (i = 3; i <= NF; i++) {
          if (i == NF && $i == "\\") sep = ""
          else        sep = ", "
          if ($i == "\\") {
            group = sprintf("%s,", group)
            next_line = 1;
          }
          else if (length(group) > 0)
            group = sprintf("%s%s%s", group, sep, $i);
          else
            group = $i;
          }
          print $2 " = group alias = " group;
        }
      }
```

Note: this script is contained in the release as *utils/mailrc.awk*.

Elm Configuration Guide

How to install and customize 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

Elm Configuration 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

This document is intended as a supplement to the *Elm Users Guide* and *Elm Reference Guide* and should be of interest mainly to people at a site installing, maintaining, and/or modifying the source code to the Elm mail system.

It is *required* that installation be done by using the *Configure* script supplied with the system. Please see the file *Instruct* for further information on running *Configure*.

The remainder of this document discusses the various questions asked by the *Configure* script and the options available via direct editing of various files and parameters. As indicated above, almost all of the sites that install Elm should find the *Configure* script more than sufficient.

Using *Configure*

Configure is a shell script that automatically determines the type of system it is running on and tunes the parameters of Elm to fit that system and its environment. Where the installer has a choice, it asks questions of the installer. *Configure* provides its own instructions when run, so they are not repeated here. However, when the installer is presented with a choice, this next section explains some of the options available. Not all the questions or options to those questions are explained here.

Use fcntl style locking?
Use flock style locking?
Use dotlock (.lock) style locking?

Elm, like all Mail User Agents (MUAs), has times when it needs exclusive access to the mail spool file. There are three methods of locking the mail spool file: *.lock* files, *fcntl*, and *flock*. *.lock* files is the original method, and is used by MUAs and Mail Transport Agents (MTAs). Whenever possible *.lock* files should be enabled to assure backwards compatibility with older MUAs and MTAs.

BSD systems introduced *flock* style locking. It uses the *flock(2)* system call to lock the file on the local node. System V later introduced *fcntl* style locking, which can also use a protocol for remote locking across the network. Where both styles are available, it is advisable to use both, unless you are sure that only one is in use at your site. Under many System V Release 4 (SVR4) systems, they both use the same underlying system call (*flock* is translated into *fcntl* style locking), so for that version of UNIX[®], *flock* is not needed and *fcntl* style alone can be used.

Enable calendar feature?

Elm has a feature to take specially marked lines within mail messages and add them to a file for use by the system calendar program. The command to do this extraction needs to be enabled to work. There is also a follow on question regarding the name of the calendar file:

Default calendar file?

which is usually *calendar* on most systems. This file resides in the user’s home directory, not their *.elm* directory.

Does your /etc/passwd file keep full names in Berkeley/V7 format (name first thing after ‘:’ in GCOS field)?

Elm uses the full name from the password file if it is available. There are two major ways this name is stored. Berkeley/V7 systems place the name as the entire GCOS field string, that is it starts directly after the ‘:’ that delimits the fields. USG (UNIX Systems Group, or AT&T) systems put the user’s name after a department number and separate it from that number by a hyphen (‘–’). The end of the user’s full name in these systems is a ‘(’. Look at your */etc/passwd* file and if either version applies, answer *yes* to this question; if neither applies, answer *no*. Elm can still get the user’s name from the *.fullname* file in their home directory.

Every now and then someone has a gethostname()/uname() that lies about the hostname but can’t be fixed for political or economic reasons. Would you like to pretend gethostname()/uname() isn’t there and maybe compile in the hostname?

Elm needs to know the correct name of the host on which it is executing to be able to create the proper headers for the outbound mail. Some systems use one name for uucp and another name for the system and others just don’t reply to the subroutines with the proper name. In this case it will be necessary to compile in the name. In all other cases this should not be needed. It is provided just in case there is a problem with your system.

Honors Content-Length: header?

Starting with SVR4, many of the MTAs are binary transparent. This allows for sending binary messages, such as encoded voice or graphics. In doing so, they no longer can tolerate changes in the message content by the mailers to aid in determining the start of the next message. To solve this problem the **Content-Length:** header was added. Elm generates the **Content-Length:** header, but to make full use of it, Elm should also not escape such sequences as ‘‘From ’’ at the start of a line of the message. If your MTA (mailer) does honor the **Content-Length:** header for determining the start of the next message, answer this question *yes*.

[®] UNIX is a registered trademark of UNIX Systems Laboratories.

Along the same lines, but now obsolete and being phased out, is a problem where an MTA thinks a message is terminated by a single lone period on a line. *sendmail* has an option to turn off this behavior and this option is set by default in *Configure*. If your mailer cannot turn off this option, add a line to *sysdefs.SH* to define the symbol **NEED_LONE_PERIOD_ESCAPE**. This symbol turns a line with a lone period into a period and a blank to avoid this problem. This symbol is ignored if the ‘‘Honors Content-Length: header?’’ question is answered *yes*.

Does your mailer understand INTERNET addresses?

Elm works with systems that can process the ‘@’ character of Internet format addresses or with the ‘!’ format of *uucp* addresses. If your MTA understands the ‘@’ format addresses, they should be used and this question answered *yes*. If messages bounce when you send them with ‘@’ format addresses (such as ‘‘elm@dsi.com’’), then answer this question *no*.

Am I going to be running as a setgid program?

On USG and many other type systems, access to the mailboxes and the mailbox directory is via the group permissions. The MUAs, such as Elm, need write access in this directory to be able to move the mailbox around for internal editing and to create lock files. If the permissions on your mailbox directory are ‘‘drwxrwxr-x’’ (no write access for ‘‘others’’), then Elm needs to be a *setgid* program.

What is the default editor on your system?

If no editor is specified in the user’s *.elm/elmrc* file, this is which editor to use. The editor is used to compose outbound mail messages.

What pager do you prefer to use with Elm?

This is the standard pager to use for reading messages. Besides the usual system pagers, two Elm specific internal options exist: *builtin* and *builtin+*. The *builtin* pager is faster to execute but much less flexible than the system provided pagers. The *builtin+* pager just clears the page before displaying the next page, otherwise the two versions are identical.

Other Configurable Parameters

The following parameters rarely need to be changed, but are provided if you need them. *Configure* does not prompt for their values. To change them, edit the *hdrs/sysdefs.h* file directly after running *Configure*.

MAX_IN_WEEDLIST	The maximum number of headers that can be specified in the weedout list of the *.elm/elmrc* file. A suggested alternative approach if this number is too small is to specify initial substrings in the file rather than increasing the number. For example, say you want to weedout the headers ‘‘Latitude:’’ and ‘‘Latitudinal-Coords:’’, you could simply specify ‘‘Latitud’’ and match them both! Furthermore you could also specify headers like ‘‘X–’’ and remove all the user defined headers!
MAX_HOPS	When using the **group reply** command, this is the maximum number of hops that a message can have taken. This is used to try to optimize the return address (remove cyclic loops and so on) and regular use should show that the default of 35 is plenty more than you’ll ever need!
system_text_file	This is the source text file for the system level aliases. See either the *newalias* man page or *The Elm Alias System Users Guide* for further details.
system_hash_file	This is the file that contains the hashed version of the system aliases generated by *newalias*.
system_data_file	This is the other file the *newalias* command installs in the system alias area and contains the actual addresses for each of the aliases contained in the hashed data file.
DEBUGFILE	The name of the file to put in the user’s home directory if they choose to use the **–d** (**debug**) option.

OLDEBUG	The name of the file to save the previous debug output as (this feature was added to ensure that users wanting to mail bug reports wouldn't automatically overwrite the debug log of the session in question).
temp_lock_dir	Directory for lock files for XENIX.
temp_file	Temporary file for sending outbound messages.
temp_form_file	A place to store temporary forms (for Forms Mode) while answering them.
temp_mbox	Place to keep a copy of the incoming mailbox to avoid collisions with newer mail.
temp_print	File to use when creating a printout of a message.
temp_edit	File to use when editing the mailbox file on XENIX.
temp_uuname	Where to redirect output of the *uuname* command.
mailtime_file	File to compare date to to determine if a given message is new since the last time the mail was read or not.
readmsg_file	File to use when communicating with the *readmsg* program (see that program for more information).
smflags	Defines the options to hand to *sendmail* if and when the program chooses to use it.
smflagsv	Defines the options to hand to *sendmail* in verbose voyeur mode.
mailer	If you don't have *sendmail*, this is the mailer that'll be used.
helpfile	The help file name prefix.
ELMRC_INFO	The file containing textual messages associated with each Elm variable setting in the user's *.elm/elmrc* file. This is used when the user chooses to save the options from within the main program.
elmrcfile	The name of the automatic control file within the *.elm* directory (by default *elmrc*).
old_elmrcfile	When a new *elmrc* file is saved, the old one is also saved by renaming it to whatever this identifier is set to.
system_rc_file	The name of the global *elmrc* file (default is *$lib/elm.rc*). This is where the system administrator puts global default values for any parameters controlled by the *.elm/elmrc* file.
mailheaders	The name of the optional file that users may have that is included in the headers of each outbound message.
dead_letter	If the user decides not to send a message it is instead saved to this filename in their home directory.
unedited_mail	In the strange case when the mailer suddenly finds all the directories it uses shut off (like */usr/mail* and */tmp*) then it'll put the current mailbox into this file in the user's home directory.
newalias	How to install new aliases (note that you MUST have the **–q** option!).
readmsg	What the *readmsg* program is installed as.

Runtime configuration — the global *$lib/elm.rc* file

No global default file is provided as part of the installation of Elm. If you wish to override any default parameters for all users, create a file as defined in the **system_rc_file** above. An easy way to create it is to copy a saved *.elm/elmrc* and edit it to remove all user-specific parameters. Of particular interest are three additional parameters you may set: **hostdomain**, **hostfullname**, and **hostname**. See **String Variables** in section 2 of the *Elm Reference Guide* for details on these variables.

When *Configure* Fails

The *Configure* script has run successfully tens of thousands of times. But maybe your system is the one in ten thousand that will confuse it. For example, *Configure* assumes that if your system has a feature it would like, then it should be used. If vendors never made mistakes, then that might be a good assumption. In reality, *Configure* might want to use a feature you'd prefer it didn't.

When *Configure* completes its examination of your system, it gives you a final chance to make changes. When *Configure* asks

If you need to edit config.sh, do it as a shell escape here:

you may briefly jump out of *Configure* to make changes. For example, if *vi* is your preferred editor, type **!vi config.sh**.

Unfortunately, *Configure* makes the same mistakes every time it runs. This means that if you ever apply updates or changes to the Elm system and rerun *Configure*, you will have to manually make the same configuration changes. Or more likely, you'll forget that the changes are required.

The solution to this problem is to create a *config.over* file in the Elm base directory. This file may contain any valid *sh* commands, and is loaded by *Configure* immediately before the *config.sh* file is created. For example, supposed that *Configure* creates a *config.sh* file with the definition

d_feature=define

but instead you want it to say

d_feature=undef

All you need to do is create a *config.over* file and put that final line in it.

The Elm Mail System

A Replacement Mailer for All Unix Systems

USENET VERSION

(2.4 Release System)

Syd Weinstein, Elm Coordinator
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

The Elm Filter System Guide

What the filter program is, what it does,
and how to use it

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

The Elm Filter System 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

One of the greatest problems with the burgeoning electronic mail explosion is that I tend to get lots of mail that I don't care about. Amusingly, perhaps, I have the equivalent of electronic junk mail. Not amusing, however, is the fact that this can rapidly accumulate and end up taking over my mailbox.

At the same time I often get mail that, while it is interesting and important, can easily be filed to be read later, without ever actually having to clutter up my incoming mailbox.

This, then, is what *filter* does! The *filter* program allows you to define a set of rules by which all incoming mail is screened, and a subsequent set of actions to perform based on whether the conditions are met or not. *Filter* also has the ability to mail a summary of what actions it performed on the incoming mail as often as you'd like.

Writing the Rules

The language for writing *filter* rules is pretty simple, actually. The fundamental structure is:

```
if  (condition)  then  action
```

Where *condition* is constructed by an arbitrary number of individual conditions of the form ‘‘*field relation value*’’ (an optional further type of rule is of the form ‘‘always *action*’’ but should only be used as the last rule in the ruleset, for obvious reasons). The *field* value can be:

subject
sender
from
to
lines

For the field **lines**, the *relation* can be any of the standard relationships (‘>’, ‘<’, ‘>=’, ‘<=’, ‘!=’ and ‘=’). **contains** is equivalent to the relation ‘=’ or you can skip the relationship entirely (e.g. ‘‘subject = "joe"’’, ‘‘subject contains "joe"’’, and ‘‘subject "joe"’’ are all equivalent). The *value* is any quoted string that is to be matched against or a number if **lines** is the *field* being considered.

The relation **matches** (or ‘˜’) takes an *egrep* like regular expression as the *value*. The regular expression must be enclosed in pairs of ‘/’. If you need a ‘/’ in your expression write it as ‘‘\ /’’.

Individual conditions are joined together by using the word **and**, and the logic of a condition can be flipped by using **not** as the first word (e.g. ‘‘subject "fun" and not subject "games"’’). We’ll see more examples of this later.

Note that the ‘‘or’’ logical conjunction isn’t a valid part of the *filter* conditional statement.

Finally, *action* can be any of:

Action	**Description**
delete	Deletes the message
save *foldername*	Saves a copy of the message in the specified foldername.
savecopy *foldername*	Does the same as save, but also puts a copy in your mailbox.
forward *address*	Sends the message to the specified address.
forwardc *address*	Sends the message to the specified address and leaves a copy in your mailbox.
execute *command*	Feeds the message to the specified command (or complex sequence of commands) as standard input.
executec *command*	Like **execute** but also leaves a copy in your mailbox.
leave	Leaves the message in your mailbox.

Note that the **execute** command assumes that you are doing something useful with incoming mail. The command does *not* automatically append the message to your mail spool. Use **executec** for that. A command such as:

```
if (from = "dave") then execute "echo mail from dave > /dev/tty"
```

will result in *losing* your mail. A more suitable use for execute would be:

```
if (from = "boss") then execute "vacation pat"
```

Foldernames can contain any of a number of macros, too, as we’ll see in the example ruleset below. The macros available for the string fields are:

Macro	**Meaning**
%d	day of the month
%D	day of the week (0-6)
%h	hour of the day (0-23)

%m	month of the year (1-12)
%r	return address of message
%s	subject of original message
%S	‘‘Re: *subject of original message*’’
%t	current hour and minute in HH:MM format
%y	year (last two digits)
%&	the string that matched the last regular expression
%1-%9	the corresponding subexpression in the last regular expression that matched

If a message has a subject of ‘‘This is a test’’, and the rule is

```
if (subject matches /([a–z]+) a ([a–z]+)/) then "echo %& %1 %2"
```

then **%&** expands to ‘‘is a test’’, **%1** expands to ‘‘is’’ and **%2** expands to ‘‘test’’.

Foldernames may contain a leading ‘‘˜/’’ which expands to your home directory.

The rules file can also contain comments (any line starting with a ‘#’) and blank lines. Metacharacters may be escaped by a preceding ‘\’ to remove any special meaning. If you need a ‘%’ character as part of an argument (such as an address), use ‘‘\%’’. If you need a double quote in a double quoted string, use ‘‘\"’’. If you need a single quote in a single quoted string, use ‘‘\´’’. If you need a backslash in any kind of string, use ‘‘\\’’.

Unless you specify an alternate file via the **–f** option, the rules file needs to reside in your *.elm* directory off your home directory and be called *.elm/filter–rules*. Here’s an example:

```
# $HOME/.elm/filter–rules
#
# Filter rules for the Elm Filter program. Don´t change without some
# serious thought. (remember – order counts)
#
# (for Dave Taylor)
# rule 1
if (from contains "!uucp") then delete
# rule 2
to "postmaster" ? save "/tmp/postmaster–mail.%d"
# rule 3
if (to "culture" and lines > 20) ? save "˜/Mail/culture"
# rule 4
subject = "filter test" ? forward "hpldat!test"
# rule 5
if [ subject = "elm" ] savecopy "˜/Mail/elm–incoming"
# rule 6
subject = "display–to–console" ? execute "cat – > /dev/console"
```

(Notice the loose syntax — there are lots of valid ways to specify a rule in the *filter* program!!)

To translate these into English:

1. All messages from ‘‘uucp’’ should be summarily deleted.

2. All mail to ‘‘postmaster’’ should be saved in a folder (file) called */tmp/postmaster–mail.n*, where *n* is the

numeric-day-of-the-week.

3. All mail addressed to ‘‘culture’’ with at least 20 lines should be automatically appended to the folder *$HOME/Mail/culture*.

4. All messages that contain the subject ‘‘filter test’’ should be forwarded to me, but via the address ‘‘hpldat!test’’ (to force a non-user forward).

5. All messages with a subject that contains the word ‘‘elm’’ should be saved in the folder *$HOME/Mail/elm–incoming* and also dropped into my mailbox.

6. Any message with the subject ‘‘display-to-console’’ will be immediately written to the console.

Notice that the *order* of the rules is very important. If we, for example, were to get a message from ‘‘uucp’’ that had the subject ‘‘filter test’’, the *filter* program would match rule 1 and delete the message. It would never be forwarded to ‘‘hpldat!test’’. It is for this reason that great care should be taken with the ordering of the rules.

Checking the rules out

The *filter* program has a convenient way of check out the rules you have written. Simply invoke it with the **–r** (**rules**) option:

```
% filter –r
Rule 1: if (from = "!uucp") then
        Delete
Rule 2: if (to = "postmaster") then
        Save  /tmp/postmaster–mail.3
Rule 3: if (to = "culture" and lines > 20) then
        Save  ˜/Mail/culture
Rule 4: if (subject = "filter test") then
        Forward  hpldat!test
Rule 5: if (subject="elm") then
        Copy  and  Save  ˜/Mail/elm–incoming
Rule 6: if (subject="display–to–console") then
        Execute "cat – > /dev/console"
```

There are a few things to notice — first off, these are the parsed and rebuilt rules, so we see them all in a consistent format. Also, notice on the filename for rule 2 that *filter* has correctly expanded the **%d** macro to be the day of the week (in this case, Wednesday, denoted by the digit ‘3’).

It is **highly** recommended that you always check your ruleset before actually using it to filter your mail!

Actually Using the Program

Now the bad news. If you aren’t running *sendmail* or some other Mail Transport Agent that supports delivery to a program you cannot use this program as currently written. Why? Because the *filter* program expects to be put in your *.forward* file (or equivalent concept for MTAs other than *sendmail*) that causes the messages to be delivered to the *filter* program. That is something that only *sendmail* and other smart MTAs look at!

The format for the entry in the *.forward* file (located in your home directory) is simply:

```
"|/usr/local/bin/filter"
```

Alright, it isn't quite *that* simple! Since *filter* is invoked by processes that don't know where you are logged in, you need to have some way to trap the error messages. For ease of use, it was decided to have all the messages written to the file specified by **–o** (or *stderr*) which means that you have two main choices for the actual entry. Either:

```
"|/usr/local/bin/filter –o /dev/console"
```

which will log all errors on the system console (each error is prefixed with ''filter (*username*)'' to distinguish it), or:

```
"|/usr/local/bin/filter –o /tmp/joe.filter_errors"
```

if you want to have a copy saved to a file. Note that the quotes are a required part of the line. A possible strategy would be to have the errors written to a file and to then have a few lines in your *.login* (or equivalent) script like:

```
if ( –f /tmp/joe.filter_errors) then
   echo  " "
   echo "Filter program errors;"
   cat /tmp/joe.filter_errors
   echo " "
endif
```

You can also use the **–v** option in combination with the above to have a more verbose log file (including action taken with date/time stamp) saved by having in your *.forward* file:

```
"|/usr/local/bin/filter –vo /tmp/joe.filter_errors"
```

Suffice it to say, you can get pretty tricky with all this!!

Summarizing the Actions Taken

The *filter* program can keep a log of all actions performed, including what rules it matched against, in your *.elm* directory in a file called *.elm/filterlog*. You can either directly operate on this file, or, much more recommended, you can use one of the two summarization options to *filter* and let *it* do the work for you!

The difference between the two is best demonstrated by example:

```
% filter –s
                Summary of Filter Activity
                --------------------------
A total of 418 messages were filtered:
The default rule of putting mail into your mailbox
         applied 364 times (87%)
Rule #1: (delete message)
         applied 1 time (0%)
Rule #2: (save in "˜/Filtered–Mail/netnews.12")
         applied 8 times (2%)
Rule #3: (save in "˜/Filtered–Mail/postmaster.12")
         applied 14 times (3%)
Rule #5: (save in "˜/Filtered–Mail/risks.12")
         applied 3 times (1%)
Rule #6: (save in "˜/Filtered–Mail/rays.12")
         applied 28 times (7%)
```

versus:

```
% filter –S
[the output as listed above, followed by:]
Explicit log of each action;
Mail from taylor about Filter Summary
         PUT in mailbox: the default action
Mail from news@hplabsz.hpl.hp.com about Newsgroup comp.editors created
         PUT in mailbox: the default action
Mail from root about Log file: cleanuplog
         PUT in mailbox: the default action
[etc.]
```

To actually use either of the summarizing options, there are two ways that are recommended. The preferred way is to have a line in either your *crontab* file (ask your administrator for help with this) that invokes the *filter* program as often as you desire with the **–s** option. For example, I have a summary mailed to me every morning at 8 o'clock:

```
0 8 * * * "/usr/local/bin/filter –s | elm –s ´Filter Summary´ taylor"
```

An alternative is to have your *.login* (or equivalent) execute the command each time you log in.

Note that if you want to have your log files cleared out each time the summary is generated you'll need to use the **–c** option too. Also, if you want to keep a long list of actions performed you can do this by saving it as you display it. A way to do this would be, if you were to have the invocation in your *.login* (or equivalent) script, to use:

```
echo "Filter Log;"
filter –c –s | tee –a PERM.filter.log
```

which would append a copy of all the output to the file *PERM.filter.log* and would avoid you having to read larger and larger summaries of what the program had done.

If you prefer not to have *filter* generate log files, use the **–q** option. Messages that are written to *stderr* or the **–o** file are still generated; only message to the log files are quashed.

Further Testing of the Ruleset

Using the *readmsg* command, it is quite easy to test the rules you've written to see if they do what you desire.

For example, we can use the **–n** option to *filter*, which means ‘‘don't actually do this, just tell me what rule you matched, if any, and what action you would have performed’’ (you can see why a single letter option is easier to type in!!), and feed it each message in our mailbox by using a command like:

```
% set message=1
% set total_messages=`messages`
% while  (1)
> if ($message > $total_messages) exit
> readmsg –h $message | filter –n
> @ message++
> end
```

which will then hand each of the messages in your mailbox to the *filter* program and display what action would have been taken with that message and why.

For example, if we do this for a few interesting messages in my mailbox, we'd end up with output like:

```
Mail from taylor about filter test
     FORWARDED to hpldat!taylor by rule;
       subject="filter test"  ? forward "hpldat!test"
Mail from bradley%hplkab@hplabsc about Re: AI-ED mailing address for HP
     PUT in mailbox: the default action
Mail from taylor about display-to-console
     EXECUTED "cat – > /dev/console"
```

(Sharp users will notice that this is exactly the same format as the longer summary listing.)

What Forwarded Messages Look Like

When a message is forwarded to another user by the *action* being specified as ‘‘**forward** *address*’’, then *filter* can generate one of two styles of message. If the message is to you, then it simply adds it to your mailbox in such a way as to ensure that the return address is that of the person who sent the message.

If not, then the message is enclosed in a message of the form:

```
From taylor Thu Oct  2 15:07:04 1986
Date: Thu, 2 Oct 86 15:06:58 pdt
Subject: "filter test"
From: The filter of taylor@hpldat <taylor>
To: hpldat!taylor
X-Filtered-By: filter, version 1.4
–– Begin filtered message ––

   From taylor Thu Oct  2 15:06:41 1986
   Date: Thu, 2 Oct 86 15:06:33 pdt
   From: Dave Taylor <taylor>
   Subject: filter test
   Just a simple test.
–– End of filtered message ––
```

The subject of the actual message is the same as the subject of the message being forwarded, but in quotes. The **From:** header indicates how the message was sent, and the **X-Filtered-By:** header identifies what version of *filter* was used.

Areas to Improve

While the *filter* program as presented herein is obviously a nice addition to the set of tools available for dealing with electronic mail, there are some key features that are missing and will be added in the future based on demand.

As I see it, the main things missing are:

1. Perhaps more *actions* available (but what?)

2. Certainly the ability to filter based on any field or combination of fields.

Warnings and Things to Look Out For

Since this is a pretty simple program, there are a few pitfalls, some of which have already been mentioned:

Order counts in the rules. Beware!

> **Matching** is pretty simple — make sure your patterns are sufficiently exclusive before having any destructive rules.

The log files can grow indefinitely unless you use the **–c** or **–q** options or you trim the files by other means.

Finally, as with the rest of the Elm mail system, I welcome feedback and suggestion on how to improve this program!!

Elm Forms Mode Guide

What Forms Mode is, how to use it to create
custom forms, how to reply to forms, and how to
use it for AT&T Mail messages

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

A Guide to Forms Mode in Elm

(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

While there are a lot of mail systems that allow the transmission of text and primitive pictures, to send and reply to more complex forms is simply not possible. Elm, however, took the forms mode implemented as part of the AT&T Mail package and has expanded on it to be a smoothly fit part of the overall mail system.

Forms mode gives you the ability to send ‘‘template’’ files to people and receive the filled-in replies.[1] Let’s look at an example right off.

Say we were going to use computer mail as a way to file defects with software. There is a certain amount of information we want to be able to collect when each report is made, and if it is in a specific format we can use programs to file the defects upon receipt.

The form we’ll try to emulate starts out looking like:

---

[1] Note that this feature assumes that the person on the other end is also using the Elm mail system, that both yourself and the person on the other end have their user levels set to something other than *Beginner* (**0**) and the **forms** variable set ON in their *.elm/elmrc* files.

Defect Reporting Form

Program: ____________________ Version: ______________
Operating System: ______________ Version: ______________

Defect Type: ________________

Date Found: __________________ By Whom: ____________________
Date Reported: ________________ Phone: ______________________

Description: ____
____
____

This form can actually be created almost exactly as listed above in the Elm mail system by using your standard editor and can then be mailed about as needed.

Let’s say that we want a bit more information, however, especially with fields like *Defect Type*, we want to list all the recommended answers. To create the actual form, we need merely to replace the underlines in the above form with spaces. The multi-line comments can simply be indicated by a ‘:’ by itself on a line:

Defect Reporting Form

Program: Version:
Operating System: Version:

(Valid Defect Types are: user-error, doc-error, fatal, other)
Defect Type:

Date Found: By Whom:
Date Reported: Phone:

Description
:

Thank you for filling in this form.

As you can see, it is quite simple to create forms!!

Now that you have an idea what we’re talking about, let’s actually officially define the system.

Forms Mode Specification

[Note that this is all taken from the document *Standard for Exchanging Forms on AT&T Mail*, Version 1.9 of 6/7/86, from AT&T.]

The forms mode is really quite simple. Simple enough that it is amazing that it hadn't been implemented before AT&T Mail came along!!

In a nutshell, each field is delimited by a ':' followed by a number of blank spaces or tabs that represent the valid size for that field. That is, if we have a line in the form like:

```
Phone (area-code):     Number:
```

The area-code field is limited to three characters and the number to nine (this is kind of hard to see with the proportionally spaced formatted copy, alas). The only exception to the rule is that a ':' by itself on a line represents a field that is as large as the user entering the data desires.

The actual form that is transmitted, in AT&T Mail parlance, is a ''SIMPLE'' forms handler message (as opposed to the ''ADVANCED'' handler). This means that it contains three sections:

The Message Header
```
[OPTIONS-SECTION]
***
[FORMS-IMAGE]
***
[RULES-SECTION]
```

Elm generates form messages with the *OPTIONS-SECTION* filled out, but ignores it when receiving mail. The filled out *OPTIONS-SECTION* is:

```
WIDTH=80
TYPE=SIMPLE
OUTPUT=TEXT
```

The *FORMS-IMAGE* section is that described above, i.e. prompting text followed by a ':', followed by spaces or tabs. The *RULES-SECTION* can contain explicit rules about the possible values of each field, but this is currently ignored by Elm, being a ''SIMPLE'' forms mode mail system.

Forms also have the header ''Content-Type: mailform'' to indicate to the mail system (either Elm or AT&T Mail) that a form is being sent.

Elm further indicates that a form has been received by having an 'F' as the status character in the header display section (instead of 'N' for new, etc).

Composing and Sending a Form

The first step to enable sending forms is to change the setting of the variable **forms** in your *.elm/elmrc* file to ON:

```
forms = ON
```

The next step is to send the message to someone using the **m** (**mail**) command, which drops you into an editor. Type in the form as indicated above, with appropriate colons and comments, and end the entry by leaving the editor.

The prompt is now:

```
Choose: E)dit msg, edit H)eaders, M)ake form, S)end or F)orget : @
```

so we choose **m — make form**. Elm then either rewrites the prompt without the ''M)ake form'' option, indicating that the form has been accepted, or indicates the problem and gives you a chance to correct it.

Once it has been accepted, simply use the **s** — **send message** — command and it's off!

Note that you cannot reply to a message with a form.

Replying to a Form

Let's reply to the form message we generated now. The header page of the Elm mail system indicates that the message is a form by having an 'F' next to it. So we use **r** to reply and the screen is immediately cleared and we're prompted, field by field, for the data requested. Each field has underscores in the input area to indicate the size field that is expected.

After answering all the questions we'll have a screen that looks like:

```
                                  Defect Reporting Form

Program:  The Elm Mail System____________________________
Version:  1.5______________________________________________
Operating System:  HP-UX_________________________________________
Version:  5.141 C_________________________________________

(Valid Defect Types are: user-error, doc-error, fatal, other)
Defect Type:  fatal____________________________________________________________________________

Date Found:  10/9/86__________________________________________
By Whom:  Dave Taylor_____________________________________
Date Reported:  10/9/86__________________________________________
Phone:  (415) 857–6887__________________________________

Description
(Enter as many lines as needed, ending with a '.' by itself on a line)
  When running it on a CPM system I cannot compile successfully.
  .
Thank you for filling in this form.
                          Choose: E)dit form, edit H)eaders, S)end or F)orget : @
```

Quite simple. Notice, however, that the order of prompting is left-to-right on each line, so the fields, *By Whom:* and *Phone:*, although placed in what seems like a logical place on the form, turn out to be confusing when filling in the received form since it isn't clear what *Phone:* is being asked for because of the intervention of the *Date Reported:* field.

The message that is actually sent out from this has the fields in a more acceptable format:

```
WIDTH=80
TYPE=SIMPLE
OUTPUT=TEXT
                            Defect Reporting Form
***

Program: The Elm Mail System                               Version: 1.5
Operating System: HP-UX                                    Version: 5.141 C

(Valid Defect Types are: user-error, doc-error, fatal, other)
Defect Type: fatal

Date Found: 10/9/86                              By Whom: Dave Taylor
Date Reported: 10/9/86                           Phone: (415) 857-6887

Description

   When running it on a CPM system I cannot compile successfully.

Thank you for filling in this form.
***
```

Comments on Forms Mode

As was said at the beginning, this way of sending about forms could prove to be very helpful and useful in a variety of contexts. On the other hand, until a more sophisticated forms language is used for the forms, this should be sufficient to embody the power of the idea.

I welcome any comments and thoughts on this system and also welcome possible enhancements.

I also gratefully thank Dale DeJager of AT&T Information Systems for sending me more information on AT&T Mail than I could possibly digest in any finite amount of time.

The Elm Reference Guide

A comprehensive list of all commands,
options and such to 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

The Elm Reference 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

There are many parts to a complex software system and **The Elm Mail System** is no different. This document describes fully all the options available in the mailer, including the command line options, the commands (in considerably more detail than in *The Elm Users Guide*) and the *.elm/elmrc* file.

To be more explicit, this document covers: a discussion of the *.elm/elmrc* file, command line options of Elm, outgoing mail processing, responses of various commands, mail archive folders, the Alias system, system aliases etc, more on the Elm utilities, and a section for expert mail users.

Without any further ado, then, let’s get this show on the road!!

2. The .elm/elmrc File

Elm, like lots of other software on the UNIX® system, has the ability to automatically read in a configuration file at each invocation. The file must be called *elmrc* and reside in the *.elm* directory located in your home directory. It can have any of the entries below, in any order. If you are missing any entries, or if you don't have an *.elm/elmrc* file, the default values (listed below for each option) or those values specified by your systems administrator in a system wide *elm.rc* file are used. Note that those options below designated with '*' can be altered using Elm via the options screen, while those designated with a '+' can be placed on the options screen using the **configoptions** variable. Also note that when you save a new *.elm/elmrc* file via the '>' command of the options screen, it is (re)created including only those options that you have changed via your original *.elm/elmrc* file or via the options screen.

The format for each line of the *.elm/elmrc* file is:

```
variable = value
```

String Variables

aliassortby*

You can have your alias display sorted by any of the following ways:

alias	Sorts according to *aliasname* for each address.
name	Sorts according to *username* for each address.
text	Presents the aliases in the order found in the *aliases.text* file.

Each of these fields can also optionally be prepended with the sequence ''reverse–'' to reverse the order of the sort. This doesn't imply anything about the order of the message in the *aliases.text* file itself and affects only their order on the display screen. The default is *name* order.

alteditor+

The editor to use when mailing to a message that already includes text, as the builtin editor cannot handle that situation. Messages that already include text are forwarded messages and replies where the original message is included in the reply. This value is not needed if the **editor** variable is not set to ''builtin''.

alternatives

This is a list of other machine/username combinations that you receive mail from (forwarded). This is used when the *group reply* feature is invoked to ensure that you don't send yourself a copy of the outbound message. The default is a list of no alternatives.

attribution

When you **forward** a message or **reply** to it, you can optionally attribute the quoted text to its original author. Defining the attribution string here allows you to indicate the form that the attribution should take. The sequence ''%s'' in the attribution is replaced by the name of the original author. Examples are:

```
attribution = According to %s:
attribution = %s writes:
```

calendar*

This is used in conjunction with the '<' *scan message for calendar entries* command, as the file to append any found calendar entries to. The default is *calendar* in your home directory.

® UNIX is a registered trademark of UNIX Systems Laboratories.

charset — This is the character set used for messages with a ‘‘Content-Type: text/plain’’ header. The default depends on your site’s installation, but is usually US-ASCII. **charset** is only recognized if MIME (Multipurpose Internet Mail Extension) support is configured. You could set **charset** to your preferred national character set, but be aware that there is currently no way to change it online. Also you must be aware that Elm probably needs *metamail* to display messages with **charset**=US-ASCII if you change **charset**. Elm tries to know which character set could display US-ASCII too, but its list of compatible character sets is probably not complete.

displaycharset — This is the character set which is supported by your terminal. The default depends on your site’s installation but is usually US-ASCII. For sites with X Window System[TM] support, ISO-8859-1 is a reasonable default.

compatcharsets — This is the list of character sets which are more or less a superset of US-ASCII. This enables Elm to display messages with **charset**=US-ASCII with the builtin pager, or your preferred pager, instead of calling *metamail*. **compatcharsets** is only recognized if MIME support is configured. The ISO-8859-X character sets are defaulted.

configoptions — This is a list of letters that indicate which of the run-time configurable options you desire placed on the options screen (see section 7, **Commands**, for the **options** command). There are 22 run-time configurable options, but only room for 15 on a 24-line screen. The default list is *ˆ_cdefsopyv_am_un*. Two additional characters can be specified for formatting. Those are ‘_’, which adds a blank line, and ‘ˆ’, which places the title message on that line instead of the bottom of the screen. The letters ‘i’, ‘q’, and ‘x’ are reserved for ‘‘return to index’’, ‘‘quit’’, and ‘‘exit’’, respectively, and are not listed as part of the **configoptions** list. The options controlled by each letter are:

a	A)rrow cursor *(arrow)*
b	B)order on copy *(prefix)*
c	C)alendar file *(calendar)*
d	D)isplay mail using *(pager)*
e	E)ditor (primary) *(editor)*
f	F)older directory *(maildir)*
h	H)old sent message *(copy)*
j	J) reply editor *(alteditor)*
k	K) pause after pager *(promptafter)*
l	A(l)ias Sorting *(aliassortby)*
m	M)enu display *(menu)*
n	N)ames only *(names)*
o	O)utbound mail saved *(sentmail)*
p	P)rint mail using *(print)*
r	R)eply copies msg *(autocopy)*
s	S)orting criteria *(sortby)*
t	T)ext editor (˜e) *(easyeditor)*
u	U)ser level *(userlevel)*
v	V)isual Editor (˜v) *(visualeditor)*
w	W)ant Cc: prompt *(askcc)*
y	Y)our full name *(fullname)*
z	Z) signature dashes *(sigdashes)*

[TM] X Window System is a trademark of the Massachusetts Institute of Technology.

easyeditor+ The editor to be used by the ‘‘~e’’ escape within the builtin editor. The default value is the value of the configuration variable *emacs_editor* (see *The Elm Configuration Guide*).

editor* The editor to use when typing in new mail. If you select ‘‘none’’ or ‘‘builtin’’ you’ll get a Berkeley Mail style interface for all mail that doesn’t already have text in the buffer (e.g. a reply, mail with a ‘‘signature’’, etc.) There are two possible formats for it, either a command that can have a filename appended to it before being executed, or a string that contains the metasequence ‘‘%s’’ which is replaced by the name of the file before being executed. Examples of each are:

```
editor = emacs –nw
editor = emacs –nw %s –f text-mode –f turn-on-auto-fill
```

The default is to use the value of $EDITOR in your current environment, and if not set, an editor selected by the person who configured Elm for your system.

escape The character used with the builtin editor (see **editor** above) to escape from text entry to input a command. When a line begins with this character, the builtin editor interprets it as a command rather than as text to add. The default is ‘~’ (tilde).

fullname* This is the name the mailer uses in messages you send. It is highly recommended that you use your full name and nothing strange or unusual, as that can appear extremely rude to people receiving your mail. The default is to use the ‘‘gcos’’ field from the */etc/passwd* file on systems that use this field to store full names, and to use the contents of the *.fullname* file in your home directory on other systems.

hostdomain This is the domain name of your system. This variable is only valid in the system-wide *elm.rc* file. It is only necessary if the value returned by the *getdomainname* system call is incorrect for your mail use or if that system call is unavailable on your system. If this variable is specified, then the **hostfullname** variable must also be specified.

hostfullname This is the ‘‘fully qualified domain name’’ of your system. This variable is only valid in the system-wide *elm.rc* file. It is only necessary if the value returned by the *getdomainname* and *gethostname* system calls are incorrect for your mail use or if those system calls are unavailable on your system. It is required if either the **hostdomain** or the **hostname** variables are used within the system-wide *elm.rc* file.

hostname This is the local node-name of your system. This variable is only valid in the system wide *elm.rc* file. It is only necessary if the value returned by the *gethostname* system call is incorrect for your mail use or if that system call is unavailable on your system. If this variable is specified, then the **hostfullname** variable must also be specified.

localsignature See **signature**.

maildir* This is your folder directory. When you specify a folder name beginning with the ‘=’ metacharacter,[1] it stands for this directory name. That is, if you save a message to folder =*stuff* the ‘=’ is expanded to the current value of **maildir**. The default is the directory *Mail* in your home directory.

---

[1] Note that ‘%’ and ‘+’ are synonymous with ‘=’ throughout Elm.

pager*

This is the program to be used to display messages. You can specify ‘‘builtin’’ or the name of any standard pager. If you use ‘‘builtin+’’, each screenfull of displayed message is ‘‘paged’’ from the top of your screen with a title line, while ‘‘builtin’’ simply ‘‘scrolls up’’ subsequent screenfulls once it has ‘‘paged’’ the first screenfull. The default is to use the value of $PAGER in your current environment, and if not set, a pager selected by the person who configured Elm for your system, quite likely ‘‘builtin+’’.

precedences

Some mail transports look at a ‘‘Precedence’’ header in outbound mail messages to determine how to deliver the message. The Elm header editing menu allows you to place a precedence on your mail messages. By default, Elm allows any value to be specified as the message precedence. This option may be used to restrict the allowed precedences to a particular list. For example, you might say:

```
precedences = special-delivery air-mail first-class bulk junk
```

Exactly what precedences your mail transport supports and what they do (if anything at all!) will vary from site to site.

The distinction between the ‘‘Precedence’’ and ‘‘Priority’’ headers is subtle: the precedence tells the mail system how to handle the message and the priority tells the recipient how important the message is. Although these are quite different things, they are often related. This option will also allow you to associate message priorities with precedences. For example, you might say:

```
precedences = special-delivery:urgent air-mail:urgent first-class bulk j
```

In this example, if you select an ‘‘air-mail’’ precedence then the message priority defaults to ‘‘urgent’’. If you select a ‘‘first-class’’ precedence then no special priority is implied. The priorities given in this field are used only if you have not already assigned a priority to your message, and even if one is assigned via **precedences** you can always go back and change it.

prefix+

When you **reply** to a message or **forward** a message to another person, you can optionally include the original message. Defining the prefix value here allows you to indicate what the prefix of each included line should be. The default is ‘‘> ’’ (specified as ‘‘>_’’ — underscore is interpreted as space) and is standard in the UNIX community.

print*

This is the command used for printing mail messages. There are two possible formats for the command, depending on whether or not the command contains the sequence ‘‘%s’’. In the first form the mail message(s) to be printed are piped to the specified command. An example of this form is:

```
print = print –formfeed
```

In the second form the mail message(s) to be printed are dumped into a temporary file, and a ‘‘%s’’ in the command string is replaced with the name of the temporary file. An example of this form is:

```
print = pr %s | lpr
```

Another significant difference between the two forms is that in the second form all output from the command is discarded, but in the first form all output is displayed on the terminal. For example, if you wish to use the ‘‘pass-through’’ feature to print on the ‘‘aux’’ port of a terminal, you must use the first form. The second form is provided for backward compatibility with old versions of Elm. It is depreciated and might be removed in a future release. The default is set by the person who configured Elm for your system.

receivedmail

This is the folder to which incoming mail is saved after you’ve read it. When you answer *no* to ‘‘Keep unread messages in your incoming mailbox?’’ or *yes* to ‘‘Store read messages in your ‘‘received’’ folder?’’, this is where the messages go. The default is ‘‘=received’’, that is, a folder called *received* in your **maildir** directory.

remotesignature

See **signature**.

sentmail*

This is the folder to which a copy of outgoing mail is automatically saved. This is only done if the **copy** boolean variable is set ON. Also note that if the **savename** boolean variable is enabled then this folder may be ignored since the program may save to a folder that has the same name as the login of the person you’re sending to. Whether or not a copy is saved, and to what folder, can be changed just prior to sending a message (see the **copy** command of the mail command sub-menu in section 7, **Commands**). The default is ‘‘=sent’’, that is, a folder called *sent* in your **maildir** directory.

shell

This defines the shell to use when doing ‘!’ escapes and such. The default is to use the value of $SHELL in your current environment, and if not set, a shell selected by the person who configured Elm for your system. Note that the ‘!’ escape is optional and may not be enabled in your version of Elm.

signature

This defines the file that is automatically appended to all outbound mail before the editor is invoked. Furthermore, if you’d like a different signature file for ‘‘local’’ mail and ‘‘remote’’ mail (remote being via other hosts), you can alternatively define two variables, **localsignature** and **remotesignature**, to have the same functionality. The default is to not have signatures appended to your messages.

sortby*

You can have your folder sorted in any of the following ways:

from	Sorts according to whom each message is *from*.
lines	Sorts *shortest* to *longest* by message.
mailbox	Leaves the messages in the order found in the folder.
received	Sorts *least recently received* to *most recently received*.
sent	Sorts *least recently sent* to *most recently sent*.
status	Sorts by *priority*, *action*, *new*, *tagged*, then *deleted*.
subject	Sorts according to the *subject* of each message.

Each of these fields can optionally be prepended with the sequence ‘‘reverse–’’ to reverse the order of the sort. This doesn’t imply anything about the order of the messages in the folder itself and affects only their order on the index screen. The default is *mailbox* order.

tmpdir

Use this if you want to define your own directory for the temporary file Elm creates while running. This is only necessary if using the system temporary directory could cause problems, such as when not all NFS clients mount the common temporary directory, or when the temporary directory is prone to being cleared periodically. The default entry of the system temporary directory is normally OK.

visualeditor+

The editor to be used by the ‘‘~v’’ escape within the builtin editor. The default value is the value of the configuration variable *vi_editor* (see *The Elm Configuration Guide*).

weedout When specifying this option, you can list headers that you *don't* want to see when you are displaying a message. This list can continue for as many lines as desired, as long as the continued lines all have leading indentation. All headers in this entry append to the default weedout list. There are two special header flags. The first, ‘‘*clear-weed-list*’’, clears the default list. The second, ‘‘*end-of-user-headers*’’, terminates the entry, in case the following lines look like they might be more headers for the list. The default **weedout** list includes the following header strings:

```
>From
Apparently-To:
Content-Length
Content-Transfer-Encoding
Content-Type:
From
In-Reply-To:
MIME-Version
Message-Id:
Newsgroups:
Received:
References:
Status:
X-Mailer:
```

Numeric Variables

bounceback This is a hop count threshold value and allows you to set up the mailer so that when you send mail more than *n* machines away, it'll automatically include a ‘‘Cc:’’ to you through the remote machine. In practice this should be very rarely used. Note that this refuses to bounce mail off an Internet address. The default is to have it set to zero, which disables the function.

builtinlines This is used to determine if the builtin pager should be used on some messages even if you would usually use an external pager program. There are two ways of determining whether the builtin pager should be used. If you want any message that is shorter than *n* lines to use the internal pager, set this variable to *n*. If you want the builtin pager to be used if the message is *m* lines shorter than the number of lines on your screen, set this variable to *–m*. Setting this variable to zero will result in the message always being sent through your external pager. This variable is used only if the pager is not set to the builtin pager. The default is –3.

readmsginc This variable modifies the display of the message ‘‘Reading in *foldername*, message: #’’, which is displayed when reading a new folder. The message count is normally updated as each message in the folder is read. If you are on a slow terminal and are reading a folder with a large number of messages, the time it takes to redraw the message count can significantly exceed the time it takes to simply read the folder.

The **readmsginc** variable controls the frequency with which the message count is updated. If this parameter is set to 50, the message count will be updated after every 50 messages (i.e., at 50, 100, 150, and so forth). The default value for this parameter is 1. If a value of less than 1 is specified for this parameter, the value is ignored, and the default value is used instead.

sleepmsg This variable modifies the time Elm waits after displaying a transient message before erasing it and continuing. It can be set to zero to suppress the wait entirely. It is in units of whole seconds.

timeout	On more advanced systems, it’s nice to start up the mailer in a window and let it sit in background until new mail arrives (see *wnewmail* for another window based program), at which point it can be brought up to the forefront of the system and read. In this case, it would be quite convenient to have the mailer internally resynchronize every so often. This option specifies the number of seconds that this occurs. This is also useful for non-windowing terminals. For example, you can leave Elm running at night (I usually do) and when you come in in the morning it’ll be all ready to read your mail! The default is a 300 second (5 minute) timeout period.
userlevel*	This is what the program uses to determine the relative level of sophistication of the user. The values are ‘0’ for a new user (the default), ‘1’ for someone familiar with Elm, and ‘2’ for experts. Some advanced features are hidden from novice users, while experts get less verbose prompt messages. The default is ‘0’.

Boolean Variables

The value assigned to boolean variables can be ‘‘ON’’ or ‘‘OFF’’ only.

alwaysdelete	Set ON to set the default answer to the ‘‘Delete messages?’’ prompt to *yes* (see the **quit** command in section 7, **Commands**, and the **ask** variable below). This default answer also applies to deletions from the alias system. The default for **alwaysdelete** is OFF.
alwayskeep	Set ON to set the default answer to the ‘‘Keep unread mail in incoming mailbox?’’ prompt to *yes*. However, if you set **alwaysstore** OFF or answer *no* to the ‘‘Store read mail in ‘‘received’’ folder?’’ prompt, it is presumed that you also want to keep your unread mail in the incoming mailbox, so the value of **alwayskeep** is ignored in those cases. See the **quit** command in section 7, **Commands**, and the **ask** and **alwaysstore** variables below for more details. The default for **alwayskeep** is ON.
alwaysstore	Set ON to set the default answer to the ‘‘Store read mail in ‘‘received’’ folder?’’ prompt to *yes* (see the **quit** command in section 7, **Commands**, and the **ask** variable below). The default for **alwaysstore** is OFF.
arrow*	Sometimes you are forced to use a slow or ‘‘dumb’’ terminal. Set ON to make the current message pointer the ‘‘–>’’ sequence rather than the inverse bar. Note that this is overridden by the ‘‘–a’’ command line option (see section 3, **Command Line Options**). The default is OFF.
ask	Set OFF to tell Elm that you’d rather not be asked ‘‘Delete messages?’’ and such each time you quit, resynchronize, change folders, or return from the alias system, but that it should just use the values of **alwaysdelete**, **alwaysstore**, and **alwayskeep** without prompting. Note that when you quit **Elm**, if you use ‘Q’ instead of ‘q’, you will never be questioned, regardless of how you have **ask** set. See the **quit** commands in section 7, **Commands**, and the **alwaysdelete**, **alwayskeep**, and **alwaysstore** variables above for more details. The default for **ask** is ON.
askcc+	Set OFF to allow sending mail without being presented the ‘‘Copies to:’’ prompt for each message. This still allows you to explicitly include addresses in the ‘‘Cc:’’ list via either the header editor or ‘‘~c’’ in the builtin editor (see section 8, **Using Elm with ‘‘editor = none’’**). The default is ON.

autocopy+	Set ON for Elm to automatically copy the text of each message replied to into the edit buffer. Otherwise you will be prompted as to whether you want the message included in your reply. See the **prefix** variable under **String Variables** in section 2 for how copied text is marked. The default for **autocopy** is OFF.
confirmappend	Set ON to make Elm ask for permission to append messages to the end of any file that already exists. Whether the file is a mail folder in the user’s mail directory or an ordinary file makes no difference. The default is OFF.
confirmcreate	Set ON to make Elm ask for permission before it creates a new file to store messages in. It makes no difference whether the new file would be a mail folder in the user’s mail directory or an ordinary file. The default is OFF.
confirmfiles	This allows you to have some last resort control over Elm when a message would be appended (by copy, save, or auto-cc) to an existing file which is not a folder in your mail directory (see the **maildir** variable under **String Variables** in section 2). Set ON to make Elm ask for permission to append a message to the end of an ordinary file, otherwise it silently adds the message to the end of the specified file whether it is a folder or not. The default is OFF.
confirmfolders	Set ON to make Elm ask before creating new mail folders in your mail directory (see the **maildir** variable under **String Variables** in section 2), otherwise it silently creates new mail folders whenever a copy of a message is going to be stored in a folder that does not already exist. See the **copy**, **savename**, and **forcename** variables below for additional information about copying messages. The default for **confirmfolders** is OFF.
copy+	Set ON to have silent copies made of all outgoing mail. Where the copy of the message is saved is determined by the **maildir** and **sentmail** string variables and the **savename** and **forcename** boolean variables. Whether a copy is saved and to which folder can also be set prior to sending a message — see the **copy** command of the mail command sub-menu in section 7, **Commands**, for details. The default for **copy** is OFF.
forcename	Set ON to force creation of folders for copies of outbound mail by the recipient name. For complete details of how to enable automatic copying of outbound messages, see the **copy** and **savename** boolean variables. The default is OFF.
forms	Set ON to enable the generation of ‘‘forms’’ type messages. See the *Elm Forms Mode Guide* for further information about mail forms.
keepempty	The mail system has a habit of deleting folders when you’ve removed everything from them. Set ON to preserve empty folders as zero-length files. Note that this option does not apply to your incoming mailbox. The default is OFF.
keypad	Set ON to indicate that you have an HP terminal and want the <*NEXT*>, <*PREV*>, <*HOME*> and <*SHIFT-HOME*> keys enabled. The default is OFF.
menu*	Set OFF to inhibit the menu display on all screen displays within Elm. Note that this is overridden by the ‘‘–m’’ command line option (see section 3, **Command Line Options**). The default is ON.
metoo	Set ON to get a copy of mail you send to a mailing list you are on, otherwise you do not get a copy of such messages. The default is OFF.

names*	Set OFF to display the primary recipients' addresses on your screen with their full names when you send a message. Set ON to display only the full names. The default is ON.
movepage	Set ON to enable commands that move through the folder by pages (see the '+', '–', <*right*>, and <*left*> keys in section 7, **Commands**) to move the current message pointer to the top of that page of messages. Set OFF to not alter the current message pointer location when moving through pages. The default is OFF.
noheader	Set ON to not include the headers of messages when copying a message into the edit buffer for replying or forwarding (see the **autocopy** variable above). The default is ON.
pointnew	Set ON to cause the current message pointer to point to the first new message in your incoming mailbox when started, instead of at message #1 of the index. This has no effect for other folders since they are not expected to have ''new'' mail. The default is ON.
promptafter+	Set ON to display a command prompt rather than the index screen when exiting from an external pager. This variable has no effect on the builtin pager. See the **pager** variable under **String Variables** in section 2 to specify which pager to use to read messages. If your external pager immediately exits when it reaches the end of the message, you should set **promptafter** ON so that the last screen of the displayed message is not immediately replaced by the index screen. If your external pager doesn't exit until you command it to, you have a choice. If you usually want to see the index screen before issuing a command, setting this variable OFF eliminates the extra keystroke needed to return to the index screen. If you usually don't need to see the index screen before issuing the next command, setting it ON allows you to enter your next command without waiting for the index screen to be redrawn. The default is ON.
resolve	Set ON to move the current message pointer to the next message on the index when a mail message is ''dealt with'' through deleting, undeleting, saving, forwarding, etc. or set OFF to leave the current message pointer unchanged. The default is ON.
savename	One of the problems with electronic mail systems is that one tends to get very large, one-dimensional (flat) files that contain lots of completely unrelated mail. Elm can use a more intelligent algorithm: for incoming mail, when you **save** or **copy** it (see section 7, **Commands**), the default folder is the login name of the person who sent you the message (changed by pressing anything other than <*return*> of course). Similarly, when sending mail, instead of just blindly saving it to the **sentmail** folder, Elm can save it to a folder that is the login name of the recipient of the mail.[2] Set **savename** ON to enable this algorithm. If **forcename** is OFF (see above), the copy is saved to that folder only if the folder already exists. In practice, this means that important people that you communicate with (those that you tend to save mail from) have folders that are actually *a recorded log of the discussion in both directions* and others (random mailings) are all stuffed in the **sentmail** folder for easy perusal and removal (see the **sentmail** variable under **String Variables** in section 2). If you always want to save copies of outgoing messages in separate folders by recipient login name, you'll want to set **forcename** ON. The default for **savename** is ON.

---

[2] When sending to a group, it's saved to the login name of the first person in the list only.

sigdashes+	Set ON to tell Elm that you wish to follow the convention of prefixing your signature with ‘‘*newline dash dash blank newline*’’. This is placed in your message before your signature file (see the **signature**, **localsignature**, and **remotesignature** variables under **String Variables** in section 2). If OFF, the signature file, if any, is placed at the end of the message without any prefix. The default is ON.
softkeys	Set ON to tell Elm that you have an HP terminal with the HP 2622 function key protocol and that you’d like to have the function keys available while in the program. The default is OFF.
titles	Set ON to have the first line of a message titled with: `Message` *N/M* `from` *username* *date* `at` *time* where all the information is extracted from the message. This is especially useful if you weed out all the headers of each message with a large **weedout** list (see the **weedout** variable under **String Variables** in section 2). The default is ON.
usetite	Set ON to enable use of the *termcap/terminfo* **ti/te** capabilities. Many terminal emulators require it (not the least of which is the OpenLook *cmdtool*). Some terminal emulators clear the screen on **te** (some *xterm*s). Set OFF to disable use of **ti/te**. Note that this is overridden by the ‘‘–t’’ command line option (see section 3, **Command Line Options**). The default is ON.
weed	Set ON to have Elm ‘‘weed out’’ certain headers from displayed messages, that is, not display them. The **weedout** variable under **String Variables** in section 2 allows you to custom define the set of headers you would like to not have displayed while reading messages. The default for the **weed** variable is ON.

For a better idea of how this all works, here’s my *.elm/elmrc* file. While looking through it, notice that you can have lots of comments and blank lines for readability and that you can also use ‘‘shell variables’’ and the ‘~’ (tilde) metacharacter for your home directory, and they are expanded accordingly when read in by the mailer. Note that this was automatically saved by the Elm program on the fly from the options screen.

```
#
# .elm/elmrc – options file for the ELM mail system
#
# Saved automatically by ELM 2.4 for Dave Taylor
#

# For yes/no settings with ?, ON means yes, OFF means no

# where to save calendar entries
calendar = ~/.Agenda

# what editor to use ("none" means simulate Berkeley Mail)
editor = none

# the character to use in the builtin editor for entering commands
escape = ~

# the full user name for outbound mail
fullname = Dave Taylor
```

```
# where to save received messages to, default file is "=received"
receivedmail = $HOME/Mail/received

# where to save my mail to, default directory is "Mail"
maildir = /users/taylor/Mail

# program to use for displaying messages (´builtin´ is recommended)
pager = builtin

# prefix sequence for indenting included message text in outgoing messages...
prefix = >_

# how to print a message (´%s´ is the filename)
print = lpr –Plw2 %s

# where to save copies of outgoing mail to, default file is "=sent"
sentmail = /users/taylor/Mail/mail.sent

# the shell to use for shell escapes
shell = /bin/csh

# local ".signature" file to append to appropriate messages...
localsignature = localsig

# remote ".signature" file to append to appropriate messages...
remotesignature =  remotesig

# do we want dashes above signatures? (News 2.11 compatibility and convention)
sigdashes = ON

# how to sort folders, "Mailbox" by default
sortby = Reverse–Received

# how to sort the alias list, "Name" by default
aliassortby = Name

# should the default be to delete messages we´ve marked for deletion?
alwaysdelete = ON

# should the default be to store read messages to the "received" folder?
alwaysstore = ON

# should the default be to keep unread messages in the incoming mailbox?
alwayskeep = ON

# should we use the "–>" rather than the inverse video bar?
arrow = OFF

# should the message disposition questions be displayed (ON) or
# auto–answered (OFF) with the default answers when we resync or change folders?
ask = ON

# would you like to be asked for Carbon–Copies information each msg?
```

```
askcc = ON

# automatically copy message being replied to into buffer?
autocopy = OFF

# threshold for bouncing copies of remote uucp messages...
# zero = disable function.
bounceback = 0

# save a copy of all outbound messages?
copy = ON

# do we want to be able to mail out AT&T Mail Forms?
forms = OFF

# should we keep folders from which all messages are deleted?
keepempty = OFF

# we´re running on an HP terminal and want HOME, PREV, NEXT, etc...
keypad = OFF

# should we display the three–line ´mini´ menu?
menu = ON
# would you like a copy of a message you send to an alias you are on???
metoo = OFF

# when using the page commands (+ – <NEXT> <PREV>) change the current
# message pointer...?
movepage = ON

# just show the names when expanding aliases?
names = ON

# when messages are copied into the outbound buffer, don´t include headers?
noheader = ON

# start up by pointing to the first new message received, if possible?
pointnew = ON

# prompt for a command after the external pager exits?
promptafter = ON

# emulate the mailx message increment mode (only increment after something
# has been ´done´ to a message, either saved or deleted, as opposed to
# simply each time something is touched)?
resolve = ON

# save messages, incoming and outbound, by login name of sender/recipient?
savename = ON

# save outbound messages by login name of sender/recipient even if the
# associated folder doesn´t already exist?
forcename = OFF
```

```
# are we running on an HP terminal and want HOME, PREV, NEXT, etc...?
# (this implies "keypad=ON" too)
softkeys = OFF

# set the main prompt timeout for resynching...
timeout = 60

# display message title when displaying pages of message?
titles = ON

# are we good at it?  0=beginner, 1=intermediate, 2+ = expert!
userlevel = 2

# enable the weedout list to be read?
weed = ON

# what headers I DON´T want to see, ever.
weedout = "Path:" "Via:" "Sent:" "Date" "Status:" "Original" "Phase"
            "Subject:" "Fruit" "Sun" "Lat" "Buzzword" "Return" "Posted"
            "Telephone" "Postal–Address" "Origin" "X–Sent–By–Nmail–V" "Resent"
            "X–Location" "Source" "Mood" "Neuron" "Libido" "To:" "X–Mailer:"
            "Full–Name:" "X–HPMAIL" "Cc:" "cc:" "Mmdf" "Network–" "Really–"
            "Sender:" "Post" "Message–" "Relay–" "Article–" "Lines:"
            "Approved:" "Xref:" "Organization:" "*end–of–user–headers*"

# alternative addresses that I could receive mail from (usually a
# forwarding mailbox) and don´t want to have listed...
alternatives = hplabs!taylor  hpldat!taylor  taylor@hplabs  taylor%hpldat
```

3. Command Line Options

There are a number of command line options to the Elm program, with only one that needs to be remembered: ‘‘–?’’or ‘‘–h’’ for help.

The options are:

–a This allows you to have the ‘‘–>’’ arrow pointer rather than the inverse bar. This can also be set in the *.elm/elmrc* file with the boolean variable **arrow**.

–c Check only. This is useful for expanding aliases without sending any mail. The invocation is similar to invoking Elm in send-only mode:

`elm –c` *list-of-aliases*

–d *n* Set debug level to *n*. Useful for debugging the Elm program, this option will create a file in your home directory called *ELM:debug.info* containing a running log of what is going on with the program. Level *n* can be 1 through 11, where the higher numbers generate more output. This option might be disabled by the the person who configured Elm for your system.

–f *folder* Read the specified folder rather than the default incoming mailbox. Note that you can use the same metacharacters (e.g. ‘=’) as when you *change folders* from within the program. You can also use the same abbreviatory symbols (‘!’, ‘>’ and ‘<’), but remember to ‘‘single quote’’ them in case they have special meaning in the shell you use.

–h or –? Help message. Gives a short list of all these options and exits.

–i *file* Include a prepared file in the edit buffer before sending. This facilitates using Elm with other programs that interface with mail (like news readers, for example). There is an example of how to set up the *rn* news reading program to use Elm in *The Elm Users Guide*. The file specified is copied into the temporary file just before the signature file.

–k Keypad enable. This option lets the Elm program know that you’re on an HP terminal, and it can then interpret the <PREV>, <NEXT> and <HOME>/<SHIFT>-<HOME> keys accordingly. If you are not on an HP terminal, it is recommended that you do NOT use this option. See also the **keypad** variable, described under **Boolean Variables** in section 2.

–K Keypad + softkeys enable. The Elm mailer can use the HP softkeys as an alternative form of input. If you specify this option be sure that you’re on an HP terminal that can accept the standard 2622 terminal escape sequences! See also the **softkeys** variable, described under **Boolean Variables** in section 2.

–m Inhibit display of the 3-line menu when using the mailer. This, of course, gives you three more message headers per page instead. See also the **menu** variable, described under **Boolean Variables** in section 2.

–s *subject* In send-only and batch mode, this is how to indicate the subject of the resulting message. Please see the section on **Non-Interactive Uses of Elm** in *The Elm Users Guide* for more information.

–t Disable use of the *termcap/terminfo* **ti/te** capabilities. Many terminal emulators require it (not the least of which is the OpenLook *cmdtool*). Some terminal emulators clear the screen on **te** (some *xterm*s). See also the **usetite** variable, described under **Boolean Variables** in section 2.

–z This causes Elm not to start if you don’t have any mail, but instead to display the message ‘‘You have no mail.’’ This emulates the behavior of programs like *Berkeley Mail*.

All the above options default to reasonable values, so there is usually no need to use them. Furthermore, the most used options are available through the *.elm/elmrc* file, described in section 2.

4. Multi-Media Capabilities of Elm

Elm optionally provides you with some Multi-Media features, which are compliant to the MIME (Multipurpose Internet Mail Extension) IETF draft.

If the support is compiled into Elm, on the receiving side Elm accesses Metamail from Nathaniel Borenstein of Bellcore. If you receive a MIME compliant message, Elm calls Metamail automatically to display the message. Metamail asks you if you want to display each part of the message and uses the display programs available at your site. This is controlled through the *mailcap* file.

On the sending side, there is a simple mechanism integrated in Elm to compose MIME compliant messages. If you have one or more key lines of the form

`[include` *file contenttype/subtype encoding*`]`

in the message body, at each of these key lines, a file is included, and becomes a part of the message. The text lines before, between and after the *include* lines go into extra message parts of type *text*.

As an example, say you want to include the file *foo.gif* into your message, which is a GIF image, and you want to use *base64* encoding, use the following line:

```
[include foo.gif image/gif base64]
```

Or you want to include a text file which contains plain ASCII:

```
[include foo.txt text/plain]
```

The *encoding* parameter is optional and the default is *7bit*.

Refer to RFC1341 for valid *contenttype/subtype* and *encoding* parameter values.

5. Special Outgoing Mail Processing

There are a few extra features that Elm offers on outgoing mail that are worthy of mention.

The first, and probably the most exciting feature,[3] is the ability to send *encrypted* mail! To do this is extremely simple: you need merely to have two key lines

```
[encode]
```

and

```
[clear]
```

in the message body.

Consider the following outgoing message:

```
Joe,
Remember that talk we had about Amy?  Well, I talked to my manager
about it and he said...
uhh...better encrypt this...the usual `key´...
[encode]
He said that Amy was having family problems and that it had been
affecting her work.
Given this, I went and talked to her, and told her I was sorry for
getting angry.  She said that she understood.
We´re friends again!!
[clear]
Exciting stuff, eh?
                                        Mike
```

---

[3] Unfortunately, at many non-US sites, it's quite probable that you won't be able to use this feature since you won't have the *crypt()* library available due to licensing restrictions.

While this is obviously quite readable while being typed into the editor, as soon as the message is confirmed as wanting to be sent, the Elm mailer prompts with:

```
Enter encryption key: @
```

and accepts a key (a series of 8 or less characters) without echoing them to the screen. After entry, it will ask for the same key again to confirm it, then **poof** it will encrypt and send the mail.

If you have the **copy** option enabled, the program will save your copy of the message encrypted too. (This is to ensure the privacy and security of your mail archive, too.)

If the mailer doesn't ask for the encryption key, it's because you don't have the *[encode]* entered as the first 8 characters of the line. It MUST be so for this to work!!

On the other end, a person receiving this mail (they must also be using Elm to receive it, since this mailer has a unique encryption program) will be reading the message and then suddenly be prompted:

```
Enter decryption key: @
```

and will again be asked to re-enter it to confirm. The program will then on-the-fly decrypt the mail and display each line as it is decoded. The *[clear]* line signifies that the block to encrypt is done.

Note that it is not possible currently to **pipe** or **print** encrypted mail.

The other feature on outgoing mail is the ability to specify what section of the message you want to have archived (assuming **copy** is enabled) and what section you don't. This is most useful for sending out source file listings and so on.

To indicate the end of the section that should be saved in the archive, you need merely to have the key line

```
[nosave]
```

or

```
[no save]
```

appear by itself on a line. This key line is removed from the outgoing mail, and indicates the last line of the message to be saved. Other than this, the saved mail is identical to the outgoing mail.

6. Customized header lines

The mailer provides a facility for including customized header lines in the messages you send. If you have an *.elm/elmheaders* file, the mailer will include its contents immediately after the regular headers of all outbound mail. The mailer supports use of the backquote convention in this file to run commands and substitute the commands' output for the backquoted text. Here's a typical *.elm/elmheaders* file:

```
Organization: Hewlett-Packard Laboratories
Phone: (415)-555-1234
Operating-System: `uname -srv`
```

These lines will be inserted after all other header lines in the message.

7. Commands

This section discusses each command in the Elm program in more detail than above, including the prompts the user can expect upon executing the command, the meaning of different options, etc.

?	Help. This command used once puts you in the *help* mode, where any key you press results in a one-line description of the key. Pressing '?' again at this point produces a summary listing each command available. Pressing '.' (period) leaves the help mode and returns you to the command level.
<space>	Display the current message. <space> is useful for reading through a mail folder. When issued from the index screen, it displays the first screen of the current message. When issued while in the builtin pager, it pages through the message to the end. When issued at the end of a message (with either the builtin pager or an external pager), it displays the first screen of the next message not marked for deletion.
<return>	Display the current message. <return> behaves somewhat differently from <space>. When issued while in the builtin pager, it scrolls the current message forward one line, and then when issued at the end of a message (with either the builtin pager or an external pager), it redisplays the first screen of the the *current* message. The latter is useful in case you have issued a non-pager command while in the builtin pager and want to restart the display of the current message.
!	Shell. This allows you to send a command to the shell without leaving the program. Note that it is possible that the person who installed Elm on your system disabled this feature.
|	Pipe. This command allows you to pipe the current message or the set of *tagged* messages through other filters as you desire. The shell used for the entire command is either the one specified in your *.elm/elmrc* file, or the default shell (see the **shell** variable under **String Variables** in section 2).
/	Pattern match. This command, at the command level, allows the user to search through all the *from* and *subject* lines of the current folder starting at the current message and continuing through the end. If the first character of the pattern is a '/', then Elm tries to match the specified pattern against *any* line in the folder. Again, this works from the current message through the end. Both searches are case insensitive.
– or <left>	Display the previous page of the message index.
+ or <right>	Display the next page of the message index.
<number><return>	Specify new current message. When you type in any digit, Elm prompts ''Set current to : *n*'', where *n* is the digit entered. Continue entering the full number and terminate with <return>. Note that changing the current message to a message not on the current page of headers results in a new page being displayed.
<	Scan message for calendar entries. A rather novel feature of the Elm mailer is the ability to automatically incorporate calendar/agenda information from a mail message into the user's calendar

file. This is done quite simply; any line that has the pattern

–> *calendar entry*

is automatically added to the user's **calendar** file when the '<' command is used (see the **calendar** variable under **String Variables** in section 2).

For example, let's say we had a message with the text:

```
Regardless of that meeting, here´s the seminar stuff:
–> 8/03 3:00pm: AI Seminar with Ira Goldstein of HP Labs
```

then using the '<' command would add the line:

```
8/03 3:00pm: AI Seminar with Ira Goldstein of HP Labs
```

to the user's **calendar** file.

a Alias. The alias system is a way by which more complex mail addresses can be shortened for the mail user. For example:

```
joe, bleu   =  Joe Bleu  =   joe@hpfcla.SSO.HP.COM
```

which allows mail to ''joe'' or ''bleu'' with the system expanding the address properly. Obviously, this saves having to remember complex addresses. A more detailed discussion can be found in either the section entitled *The Alias System* in this document or *The Elm Alias System Users Guide*.

b Bounce mail. This ''remails'' mail to someone else in such a way as to make the return address the original sender rather than you. The **forward** command is similar, but it makes the return address *you* rather than the original sender.

C Copy to folder. This command copies the current message or set of tagged messages to a folder. If there is anything in the folder currently the message is appended to the end, otherwise the folder is created containing only the newly copied messages. The prompt for this command is ''Copy to folder: ''. A response of <return> cancels the command and returns the user to the command prompt. The usual filename metacharacters are available, too. That is, this command expands filenames with '˜' (tilde) to your home directory and '=' to your **maildir** directory, if defined. This command also allows you to use '>' for your **receivedmail** folder, '<' for your **sentmail** folder, '.' for the last folder you saved or copied a message to and ''@alias'' for the default folder for ''alias''. If you use a shell wildcard in the file or folder name, you are given a list of all files or folders which match the wildcard. Elm uses your shell to find the names, so whatever wildcards you are used to will work. Finally, you can also enter '?' at the prompt to get detailed help.

c Change folder. Specifying this command allows the user to change the folder that is currently being read. This is intended for perusal and reply to previously archived messages. The prompt is ''Name of new folder: '' and entering <return> cancels the operation, while entering a filename causes the program to read that file as the new folder, if possible. This command expands filenames with '˜' (tilde) to your home directory and '=' to your **maildir** directory, if defined. This command also allows you to use '!' as an abbreviation for you incoming mailbox, '>' for your **receivedmail** folder, '<' for your **sentmail** folder, '.' for the last folder you saved or copied a message to and ''@alias'' for the default folder for ''alias''. If you use a shell wildcard in the file or folder name, you are given a list of all files or folders which match the wildcard. Elm uses your shell to find the names, so whatever wildcards you are used to will work. Finally, you can also enter '?' at the prompt to get detailed help.

d, u Delete and undelete. Neither of these two commands have any prompts and indicate their action by either adding a 'D' to the current message index entry (indicating deletion pending) or removing the 'D' (indicating that the message isn't set for deletion).

<control>-D

This command allows you to easily mark for deletion all messages that have a specific pattern. After **<control>-D** is pressed, Elm prompts for the string to match in either the *from* or *subject* lines of the messages.

<control>-U

This is the direct opposite command to the **<control>-D** command — all messages that match the specified pattern have any mark for deletion removed by this command.

e Edit mailbox. This allows you to modify the current mail file at a single keystroke. This is mostly useful for editing messages before saving them. Modifying headers should be done with extreme caution, as they contain routing information and other vital stuff for full functionality. This command may be disabled by whoever configured your Elm installation.

f Forward. Allows the user to forward the current message to another user. This copies the message into the edit buffer and allows the user to add their own message too. The prompt is ''Forward to:'' and will expand an alias if entered. See also **bounce**, above.

Elm will ask you if you want to edit the message before sending it. If you answer *yes*, Elm will prepend your prefix string to each line of the message, and let you edit the result. If you do not want the prefix string on each line, answer *no*; you will have another chance to edit the message when you get to the ''send'' menu. See the **prefix** variable under **String Variables** in section 2.

g Group reply. Identical to **reply** below, except that the response is mailed to *all recipients* of the original message except yourself. See the **alternatives** variable under **String Variables** in section 2.

h Display the current message with all headers intact. When you display a message with other commands, certain header lines are formatted and others discarded according to the **weedlist** variable, described under **String Variables** in section 2.

i Return to the index screen, when issued in the builtin pager or at the end of a message with either the builtin pager or an external pager.

j or <down>, k or <up>

These four keys work similarly to what they would do in *vi* or any of the other (precious few) screen oriented programs. The **j** and **<down>** keys move the current message pointer down to the next message skipping over any marked deleted (going to the next page if necessary) and the **k** and **<up>** keys move the current message pointer back to the previous message skipping over any marked deleted (also changing pages if necessary).

J, K These two keys work similarly to their lower case counterparts, except that they don't skip over deleted messages.

l Limit. This feature allows you to specify a subset of the existing messages to be dealt with. For example, let’s say we had a folder with four hundred messages in it, with only four or five different subjects. We could then limit what we’re dealing with by using the **limit** command. Pressing **l** would result in the prompt:

```
Criteria:
```

to which we could answer ‘‘subject *string*’’, ‘‘from *string*’’ or ‘‘to *string*’’. In our example, we could use ‘‘subject programming’’ as a criterion for selection. Once we’ve limited our selections, the screen is rewritten with just the selected messages and the top line changes to have a message like:

```
Folder is "=elm" with 92 shown out of 124 [Elm 2.4]
```

We can further limit selections by using the **limit** option repeatedly to enter further criteria.

To clear all the criteria and get back to the ‘‘regular’’ display, simply enter ‘‘all’’ as the limiting criteria. It should be noted that the selection based on ‘‘to’’ isn’t fully implemented for this version, so it is recommended that users stay with ‘‘subject’’ and ‘‘from’’ as the basis for their criteria.

m Mail. Send mail to a specified user. The prompt associated with this command is ‘‘Send mail to: ’’. Entering an alias name results in the full address being rewritten in parenthesis immediately. This prompt is followed by ‘‘Subject: ’’ which allows the user to title their note. The final prompt is ‘‘Copies to: ’’, which allows other people specified to receive ‘‘carbon copies’’ of the message, but see the **askcc** variable under **Boolean Variables** in section 2. Upon entering all three items the editor is invoked and the message can be composed.

n Next message that is not marked for deletion. Useful for displaying successive messages in a folder. When issued from the index screen, it displays the current message, and when issued while in the builtin pager or at the end of a message (with either the builtin pager or an external pager), it displays the first screen of the next message not marked for deletion.

o Options. This full-screen display allows you to alter the settings of a number of parameters, including the current sorting method, the method of printing files, the calendar file, the save file, and so on. It’s self-documenting (where have you heard *that* before?) so isn’t explained in too much detail here. See the **configoptions** variable under **String Variables** in section 2.

p Print. This allows you to print out the current message or the tagged messages to a previously defined printer. See the **print** variable under **String Variables** in section 2.

q Quit. If you in the pager, you are returned to the index screen. If you are at the index screen, Elm quits altogether. However, if you have the **ask** variable set ON, Elm first prompts you for the disposition of the messages in the current folder. If any messages are marked for deletion, it asks if you want them deleted. If the current folder is your incoming mailbox, you are also asked if read messages should be stored in your **receivedmail** folder, and if unread messages should be kept in the incoming mailbox. The default answers to these questions are set by the *.elm/elmrc* variables **alwaysdelete**, **alwaysstore**, and **alwayskeep** (see **Boolean Variables** in section 2). However, if you elect to not store your read messages (i.e. keep them) it is presumed you want to keep your unread messages, too.

Q Quick quit. This behaves similar to the **quit** command except that you are never prompted for answers to the message disposition questions. Elm disposes of messages according to the values you have set for **alwaysdelete**, **alwaysstore**, and **alwayskeep** in your *.elm/elmrc* file (see **Boolean Variables** in section 2).

r Reply. Reply to the sender of the current message. If the **autocopy** variable is set to OFF in your *.elm/elmrc* file, Elm prompts ‘‘Copy message? (y/n)’’, to which you can specify whether or not a copy of the source message is to be copied into the edit buffer. If copied in, all lines from the message are prepended with the **prefix** character sequence specified in your *.elm/elmrc* file (see **String Variables** in section 2).

s Save to folder. This command is like the **copy** command, except that the saved messages are marked for deletion and, if you are saving just the current message, the current message pointer is incremented afterwards (see the **resolve** variable under **Boolean Variables** in section 2). This command expands folder names with ‘˜’ (tilde) to your home directory and ‘=’ to your **maildir** directory, if defined. This command also allows you to use ‘>’ for your **receivedmail** folder, ‘<’ for your **sentmail** folder (see **String Variables** in section 2), ‘.’ (period) for the last folder you saved or copied a message to and ‘‘@alias’’ for the default folder for ‘‘alias’’.

t Tag. Tag the current message for a later operation.[4]

T Tag and move to next undeleted message. This command is like the ‘Tag’ command but also increments the current message pointer to the next undeleted message.

<control>-T
Tag all messages containing the specified pattern. Since *tagging* messages can occur on screens other than the one being viewed, Elm first checks to see if any messages are currently *tagged* and ask you if you’d like to remove those tags. After that, it will, similar to the **<control>-D** command, prompt for a pattern to match and then mark all messages that contain the (case insensitive) pattern in either the *from* or *subject* lines.

x Exit. This leaves Elm and discards any changes to the mailbox. If changes are pending (such as messages marked for deletion) you are asked to confirm discarding the changes. If confirmed, no messages are deleted and the status of all messages is unchanged. That is, any messages that were new will remain new instead of being noted as old, and any messages that were read for the first time will be again noted as unread.

X Exit immediately. This leaves Elm in the quickest possible manner without even prompting about discarding the changes to the mailbox. No messages are deleted and the status of all messages is unchanged. That is, any messages that were new will remain new instead of being noted as old, and any messages that were read for the first time will be again noted as unread.

When you are about to send a message with the **forward**, **mail**, or **reply** commands (see above), a small menu of the following options appears:

c Specify the folder for saving a copy of the message. This allows you to override the **copy**, **forcename** and **savename** variables from your *.elm/elmrc* file (see **Boolean Variables** in section 2). It prompts you for the name of the folder where a copy of the outgoing message is to be saved. The default displayed is taken from those three *.elm/elmrc* options and can be changed. This command

---

[4] Currently only **copy**, **pipe**, **print**, and **save** support this.

also allows you to use ‘>’ for your **receivedmail** folder and ‘<’ for your **sentmail** folder (see **String Variables** in section 2), and ‘=?’ to mean ‘‘conditionally save by name’’ and ‘=’ to mean ‘‘unconditionally save by name’’. Since you could next enter the **edit headers** command and change the recipients of your message, the name of the folder under the two ‘‘save by name’’ options is not established until you enter the **send** command. If you use a shell wildcard in the file or folder name, you are given a list of all files or folders which match the wildcard. Elm uses your shell to find the names, so whatever wildcards you are used to will work. You can also enter ‘?’ at the prompt to get help about saving.

e	Edit message (or form). Entering this command allows you to edit the text of your message or form.
f	Forget. This gets you out of sending a message you started. If you are in send-only mode, the message is saved to the file *Canceled.mail* in your home directory. Otherwise it can be restored at the next **forward**, **mail**, or **reply** command during the current session of Elm After issuing one of those commands you will be prompted with ‘‘Recall last kept message?’’
h	Edit headers. This puts you into the *header editing mode*, whereby you can edit any of the various headers of your message. Like the options screen, it’s self-documenting, so it isn’t explained in too much detail here.
i	Run *ispell* (or some other configured spelling correction program). The outgoing message is run through an interactive spelling correction program if one is available. The default spelling program is the GNU *ispell* program unless changed by the person who installed Elm on your system.
m	Make form. This converts the message you have edited into a form. See the **forms** variable under **Boolean Variables** in section 2 and *The Elm Forms Mode Guide* for more details.
s	Send. This sends the message as is without any further ado.

8. Using Elm with ‘‘editor = none’’

The Elm program also supports a builtin editor for simple message composition that is very (very) similar to the simple line editor available from the *Berkeley Mail* system.

To access it, you need merely to specify ‘‘**editor=none**’’ in your *.elm/elmrc* file. With that, any messages to be composed that don’t already have text in the buffer (e.g. no reply with the text included, etc.), will use this editor.

From the builtin editor, the following options are available for use. Each command here is prefixed with a ‘~’ (tilde). You can specify a different ‘‘escape’’ character in your *.elm/elmrc* file, if you desire (see the **escape** variable under **String Variables** in section 2).

~?	Print a brief help menu.
~b	Change the Blind-Carbon-Copy list.
~c	Change the Carbon-Copy list.

˜e Invoke the **easyeditor** editor on the message, if possible (see the **easyeditor** variable under **String Variables** in section 2).

˜f Add the specified message or current message.

˜h Change all the available headers (To, Cc, Bcc, and Subject).

˜m Same as **˜f**, but with the current **prefix** (see the **prefix** variable under **String Variables** in section 2).

˜o Invoke a user specified editor on the message.

˜p Print out, on the screen, the message as typed in so far.

˜r Include (read in) the contents of the specified file.

˜s Change the Subject line.

˜t Change the To list.

˜v Invoke the *vi* visual editor on the message.

˜< Execute the specified UNIX command, entering the output of the command into the editor buffer upon completion. For example, ‘‘˜< who’’ includes the output of the *who* command in your message.

˜! Execute a UNIX command if one is given (as in ‘‘˜!ls’’) or give the user a shell, either from the **shell** variable setting in the *.elm/elmrc* file or the default (see the **shell** variable under **String Variables** in section 2).

˜˜ Add a line prefixed by a single ‘˜’ character.

A useful note is that the **˜f** and **˜m** commands invoke the *readmsg* command, so you can pass parameters along too. For example, if we wanted to include a message from Joe, without any headers, but with each line prefixed, we could use:

```
˜m –n Joe
```

to accomplish the task.

To learn more about how they work, try ’em!

9. The Alias System

As mentioned previously, there exists in the Elm system a set of aliases that associate an arbitrary word (such as a persons name) to a complex address or group. The advantages are readily apparent; rather than remembering an address of the form:

```
host1!host2! ... !hostN!user
```

the user merely has to remember a single word.

Two alias tables are available for a each user within Elm, namely the system alias file and the user's alias file. The system alias file is created and maintained (by the system administrator) by editing the file name defined for *SYSTEM_ALIASES* in the *sysdefs.h* file (see *The Elm Configuration Guide*) and as described in the documentation with the *newalias* command, then running the *newalias* program.

An individual user can also have an alias file which works in conjunction with the system aliases. To do this, one merely needs to enter the alias menu system and create aliases with the **a** (**alias current message**) or **n** (**make new alias**) commands. Alternatively, the user can peruse the documentation for the *newalias* command and create a file as indicated therein. After executing the program, the aliases are available for use from within Elm.

Please refer to *The Elm Alias Users Guide* for complete details.

Within Elm, however, the alias system acts as an entirely different program, with its own display, own commands, and own mini-menu. The aliases are presented in a list similar to the index screen with the following menu:

```
              Alias commands:  ?=help, <n>=set current to n, /=search pattern
        a)lias current message, c)hange, d)elete, e)dit aliases.text, f)ully expand,
           l)imit display, m)ail, n)ew alias, r)eturn, t)ag, u)ndelete, or e(x)it

    Alias: @
```

The commands are:

? Help. This command used once puts you in the *help* mode, where any key you press will result in a one-line description of the key. Pressing '?' again at this point produces a summary listing each command available. Pressing '.' (period) leaves the help mode and returns you to the alias command prompt.

<space> or <return> or v
Display the current alias address. The alias address is displayed below the alias menu. This command allows you to verify the address for a person or the contents of a group alias.

$ Resynchronize the alias text file (*$HOME/.elm/aliases.text*) and alias database by rebuilding the database from the text file by running *newalias*. Aliases marked for deletion are removed, tagged aliases are untagged, and new and changed aliases are recognized. The alias screen is updated to reflect these changes.

/ Pattern match. This command allows the user to search through all the *alias* and *username* entries in the alias list starting at the current alias and continuing through the end. If the first character of the pattern is a '/', then Elm also includes the *comment* and the fully expanded *address* fields in the search. The search is case insensitive. This allows the user to find a specific alias in the situation where there are a large number of aliases.

a Alias current message. This allows the user to create an alias that has the return address of the current message as the address field of the alias. It prompts for a unique alias name. If the alias name is not unique, you will be asked if you wish to replace the existing alias. For further information, please see *The Elm Alias System Users Guide*.

c Change current alias. This will prompt for changes to the current names and address. If other aliases are tagged you will be asked if you want to create a group alias from the tagged aliases. The original alias is replaced with the new information in your individual alias file (*$HOME/.elm/aliases.text*) and then added to the database (at the next alias resync). Aliases that have been changed are marked with an 'N' (for new) until the database is updated.

d, u	Delete or undelete an alias. This allows the user to mark an alias for deletion in the same fashion as on the index screen. The deletions are not actually made until the user returns to the main menu with the **r**, **q**, or **i** commands or resyncs the display with the **$** command. Deletions on system aliases are not allowed. These commands (plus the **<control>-D** and **<control>-U** versions) behave identically to their index screen counterparts (see section 7, **Commands**).
e	Edit the *.elm/aliases.text* file. The user alias file is edited using the editor defined in the **editor** variable in your *.elm/elmrc* file (see **String Variables** in section 2). *newalias* is run after the edit.
f	Display fully expanded alias. The currently selected alias is fully expanded and displayed to the user. This is most useful when working with group aliases.
l	Limit the display. You can limit the display by alias type (person/group or user/system) or by search pattern on name or alias. Otherwise, this works exactly like the limit command on the index screen.
m	Send mail to the current alias. The user is prompted to compose a new mail message to be sent to the person or group specified by the selected alias. If aliases are tagged the message is mailed to the person(s) and/or group(s) specified by the tagged aliases. Tags are cleared after mailing the message.
n	Make a new user alias. This prompts for a unique alias name and then for an address. If the alias name is not unique, you are asked if you wish to replace the existing alias. If aliases are tagged you are asked if you want to create a group alias from the tagged aliases. The information provided is added to your individual alias file (*$HOME/.elm/aliases.text*) and then added to the database at the next alias resync.
r or q or i	Return. Return to the index screen of the Elm program. Any pending deletions are processed and newalias is run to update the database. New additions are handled at this time as well.
R or Q or I	Quick return. This behaves like the 'r' command except that you are never prompted for answers to alias disposition questions. Elm disposes of aliases according to the value set for the **alwaysdelete** variable in your *.elm/elmrc* file (see **Boolean Variables** in section 2).
t	Tag. Tag the current alias for a later operation.[5] This command (plus the **<control>-T** version) behaves identically to its index screen counterpart (see section 7, **Commands**).
x	Exit alias menu. Exits the alias menu without processing any deletions. Aliases marked for deletion are unmarked and *newalias* is not run, even if alias additions have been made.

Additionally, the movement keys ('j', 'k', '+', '–', etc.) work in the same fashion as on the index screen (see section 7, **Commands**).

---

[5] Currently only **mail**, **change**, and *n3w alias* support this.

10. Elm and Signals

Elm can handle several standard UNIX *signals* to do some special processing. Signals are interrupt messages sent from one program to another. No detailed messages are sent, but a properly configured program (such as Elm) can watch for and handle these signals.

In particular, Elm watches for the following signals and takes these actions:

ALRM	This is the alarm clock signal or time warning. Elm uses this to wake itself up periodically and check for new mail.
HUP	This is the hangup notice. It means that the terminal/modem/whatever which you have been using with Elm has become detached from the system where Elm was running. When Elm gets this signal, it aborts all the pending operations and exits, leaving your mailbox unchanged.
USR1	This is the first user-defined signal. When Elm gets this signal, it receives any pending mail, performs all the pending operations (deletes), and exits leaving all unread mail marked as new. This is the same as giving both the **$** and **X** commands.
USR2	This is the second user-defined signal. When Elm gets this signal, it receives any pending mail, performs all the pending operations (deletes), and exits, leaving all unread mail marked as old. This is the same as giving both the **$** and **Q** commands.

You would only use these signals yourself under the most unusual circumstances. For example, suppose you were using **Elm** to read mail on *host_1*. You have many messages, most of which have been read and filed (and therefore deleted), or simply deleted. You have not yet resynchronized your mailbox (**$** command). Now you go to lunch. On your return, you're stopped at the door and told to take care of an emergency. You go to another part of the building, and want to read your mail to see what the emergency is — but you can't, because Elm is still running at your desk.

What you really want is one of the scenarios given in the description of *HUP*, *USR1* or *USR2*. Use your local *ps* command to find out what the *process number* of your Elm session is. Then give the command

`kill –`*XXX process_number*

where *XXX* is either ‘‘HUP’’, ‘‘USR1’’ or ‘‘USR2’’ and *process_number* is the process number for your remote Elm session. Your remote session will terminate with the actions noted above.

11. Expert Mail Users and Debugging the Mailer

There are some additional facilities available in the Elm mailer for those people who are knowledgeable about mail protocols, or trying to debug/track down a problem.

The **h** (**display with headers**) command at the command prompt displays the current message ignoring the current setting of the **weed** variable (see **Boolean Variables** in section 2). This is most useful for answering questions of the form ‘‘I wonder what this guy put in his header?’’ and such. This command does not show up on the mini-menu because it is somewhat esoteric, but it does appear on the help screen.

The **@** command at the command prompt outputs a screen of debugging information, including the number of lines and offsets of each of the messages in the current mailbox.

The **#** command at the command prompt displays the entire stored ‘‘record structure’’ for the current message.

The **%** command displays the full computed return address of the current message.

Starting up Elm with the **–d** (debug) option (see section 3, **Command Line Options**) creates a file called *ELM:debug.info* in your home directory which contains a wealth of useful information (to me, at least!) to aid in tracking down what errors are occurring and why.

If there are any problems with Elm, please try to recreate the error with the debug option enabled and set to the highest level (11) before sending defect reports my way.

One final note: all error names reported by the program are documented in the *AT&T System V Interface Definition Reference Manual* in **errno**(2).

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.

NAME

answer - phone message transcription system

SYNOPSIS

answer [**–pu**]

DESCRIPTION

Answer is a part of the **Elm** mail system and is designed for secretaries and the like to allow them to answer phones and|or listen to phone message machines and quickly and easily transcribe the messages into electronic mail.

The program will use standard mail addresses or aliases included in the standard .elm directory alias files. It can also use its own alias set (usually a subset of the system aliases, with more fields per user (i.e. unique first names, unique last names, first-initial last name, and so on)). Note that if you want to use this capability, it means that you must create this alias set and then install it, using *newalias,* as a separate alias table for the account this program is to be run from.

Options are:

-p	Prompt for phone slip type message fields.
-u	Allow names that are not in the alias table. Without this option all names must be resolvable as aliases and answer will reprompt for a new name if the alias cannot be resolved.

Interface-wise, the program is a breeze to use - type 'answer' to start it up and then merely type a user name at the

> Message to:

prompt, fill in the requested information (simply type an "x" next to the pertinent information - "TELEPHONED", "PLEASE CALL", "URGENT", etc. if the -p option was given), and then enter a message, if any, ending with a blank line. If you elect to use the special alias file, names can be entered as 'Firstname Lastname' and the system will try to find a mail alias based on the first letter of the first name plus the last name. For example, 'Dave Taylor' will cause the program to search for an alias 'd_taylor' in the file (note the transliteration into lower case). The program will loop until killed or quit.

Answer can be easily exited by typing any of 'quit', 'exit' 'done', or 'bye' at the 'Message to:' prompt.

AUTHOR

Elm Development Group

SEE ALSO

The Elm Alias System Users Guide

newalias(1L), checkalias(1L), mail(1), mailx(1), from(1L), printmail(1L)

FILES

$HOME/.elm/aliases.text	individual alias source
$HOME/.elm/aliases.dir	individual alias dbz directory table
$HOME/.elm/aliases.pag	individual alias dbz hash table
$HOME/.elm/aliases	individual alias data table
/tmp/snd.$$	edit buffer for outgoing mail

BUG REPORTS TO

Syd Weinstein elm@DSI.COM (dsinc!elm)

COPYRIGHTS

NAME

autoreply - Automatic mail reply system

SYNOPSIS

autoreply
autoreply off
autoreply *filename*

DESCRIPTION

Autoreply is a part of **The Elm Mail System** and is designed to enable users to easily set up an automatic response to all mail they receive. This is typically during a vacation or some other period of time that the user is not available to reply to mail (although some people get enough mail that they have it permanently running to simply send an "ack" to each message)

The system is quite easy to use... You can type "autoreply" and be informed of your current status in the system (on or off), or "autoreply off" will turn your autoreplying off.

To add yourself to the system, you need merely to compose a form response in a file somewhere, then type "autoreply *file*" where *file* is the name of the file.

SPECIAL NOTE

For this to function correctly, the program "arepdaemon" must be running as a daemon in background. This program automatically checks all the currently enrolled users for new mail in their mailboxes...

AUTHOR

Elm Development Group

SEE ALSO

The Elm Users Guide

elm(1L), mail(1), mailx(1)

BUGS

Unfortunately, this program isn’t too brilliant about identifying mail that shouldn’t be responded to (like mail from other daemons, or your mother) but that’s a more general problem anyway.

Also, occasionally a user will remove themselves from the autoreply system and the system won’t turn their autoreply off. This is a problem that hasn’t been solved yet (obviously since it’s in the "bugs" section!!) and any further information would be appreciated!

BUG REPORTS TO

Syd Weinstein elm@DSI.COM (dsinc!elm)

COPYRIGHTS

NAME

checkalias - check to see if an alias is defined.

SYNOPSIS

checkalias alias, alias, ...

DESCRIPTION

Checkalias checks the user file first, and then the system alias file to try to find the specified alias or aliases. If found, the value of the alias is printed out, otherwise an error is generated. Note: this is in fact a one-line shell script;

```
elm -c $*
```

AUTHOR

Elm Development Group

SEE ALSO

elm(1L), elmalias(1L), newalias(1L), mail(1), mailx(1)

BUGS

Note that the precedence of aliases is user file then system file. This means that a user can ’overload’ an alias by having one defined in the system file also defined in theirs. This shouldn’t turn out to be a problem, but is something for the system administrator to keep in mind when creating the system alias file.

BUG REPORTS TO

Syd Weinstein elm@DSI.COM (dsinc!elm)

COPYRIGHTS

NAME

elm - an interactive mail system

SYNOPSIS

elm [**–achkKmrtwz**] [**–f** *alternate-folder*] [**–d** *debug-level*]
elm [**–s** *subject*] *list of aliases or addresses*

DESCRIPTION

Elm is an interactive screen-oriented mailer program that supersedes *mail* and *mailx.*

There are three main ways to use the **elm** mailer. One way to use the mailer is to specify a list of addresses on the command line when the mailer is invoked. This will allow sending of a single message to the specified recipients with all the options usually available in the **elm** system itself. For example, the command

```
elm -s testing joe
```

would initiate the sending of a message to the alias joe with the subject "testing" indicated and would put you into an editor for you to enter the text of the message. If the subject flag and argument were not present, you would be prompted for a message subject.

The second way, used most commonly when transmitting files and such, is to specify the subject of the message and the recipients using the command line and redirect a file as standard input. For example, the command

```
elm -s testing joe < test.c
```

would mail a copy of the file test.c to alias joe, with the subject "testing" indicated. The subject flag and argument are optional.

The third way, elm is invoked with neither subject nor addresses specified on the command line and without standard input redirected. You can then read any messages in your incoming mailbox (or specified folder), reply to messages in the mailbox, mail messages, etc.

Options are:

-a	Arrow - force the arrow cursor (instead of the inverse bar)
-c	Checkalias - expand the following aliases and return.
-d *level*	Debug - set specified debug level - Output to "$HOME/ELM:debug.info"
-f *alternative-folder*	Folder - read folder (specified) rather than the incoming mailbox.
-h	Help - give a list of starting options.
-i *file*	Include - include prepared file in edit buffer for sending.
-?	Synonymous with the "-h" option.
-k	Keypad - force knowledge of HP terminal keyboard, to allow the use of the NEXT, PREV and HOME/SHIFT-HOME keys.
-K	Keypad + softkeys - enable use of softkeys on HP terminals only.
-m	Menu off - Use the extra lines for more message headers.
-s *subj*	Subject - specify subject for message to mail.
-t	tite - turn off usage of the termcap/terminfo ti/te sequence.
-v	Version - print version and configuration information.
-z	Zero - don’t enter **Elm** if no mail is pending.

AUTHOR

Elm Development Group

SEE ALSO

The Elm documentation package, including: *The Elm Users Guide*, The Elm Reference Guide, The Elm Alias System Users Guide, The Elm Forms Mode Guide, The Elm Filter Guide, The Elm Configuration Guide

checkalias(1L), frm(1L), mail(1), mailx(1), newalias(1L), printmail(1L), readmsg(1L)

FILES

/usr/local/lib/elm-help.*	help files
/usr/local/lib/aliases.text	system alias source
/usr/local/lib/aliases.dir	system alias dbz directory table
/usr/local/lib/aliases.pag	system alias dbz hash table
/usr/local/lib/aliases	system alias data table
/usr/local/lib/aliases.hash	system alias hash table
/usr/local/lib/aliases.data	system alias data table
$HOME/.elm/aliases.text	user alias source
$HOME/.elm/aliases.dir	user alias dbz directory table
$HOME/.elm/aliases.pag	user alias dbz hash table
$HOME/.elm/aliases	user alias data table
$HOME/.elm/elmrc	customized mailer parameters
$HOME/.elm/elmheaders	customized message headers
/tmp/snd.$$	outgoing mail edit buffer
/tmp/mbox.*logname*	temporary mailbox
$HOME/ELM:debug.info	Debug output if turned on

DIAGNOSTICS

Should know about keyboards/softkey terminals other than HP. (If only termcap were that powerful!)

BUG REPORTS TO

Syd Weinstein elm@DSI.COM (dsinc!elm)

COPYRIGHTS

NAME

fastmail - quick batch mail interface to a single address

SYNOPSIS

fastmail [-b bcc-list] [-c cc-list] [-C comments] [-d] [-f fromname] [-i msg-id] [-r replyto] [-R references] [-s subject] filename|- address-list

DESCRIPTION

Fastmail is a low-level interface to the mail system that allows batch processing of mail. It's intended for mailing to very large groups of people in a staggered fashion.

The starting options are;

-b bcc-list	This allows a list of people to receive blind-carbon copies, or BCCs, of the message. This list should be full email addresses.
-c cc-list	This allows a list of people to receive carbon copies, or CCs, of the message. This list should be full email addresses.
-C comments	This allows a Comments: line to be added to the RFC822 header with any text you desire.
-d	Debug. This is helpful for strange, unfriendly errors from the program (etc).
-f from	This overrides the users name in the From: line, so that if the user was x@y, and their name was MrX then the default From: line would be "From: x@y (MrX)". Using "-f Joe" when invoking this, though, would change it to "From: x@y (Joe)"
-i msg-id	This allows a message-id to which this message refers.
-r replyto	Occasionally, you might send mail but want the replies to go to a different address (very common with mailing lists). There is a header for this purpose called "Reply-To:" which can be utilized by using this starting option. For example, we could send mail with a reply-to to list-request by using "-r list-request". The header generated would then be of the form "Reply-To: list-request".
-R references	This allows descriptive/reference text for this message.
-s subject	The subject of the message is specified by using this starting option.

Either the name of the file containing the message, or a - to indicate usage of standard-in is required.

EXAMPLE

Let's say we're user "big" on machine "big-vax" and we have a shell script called 'batch-mail' that contains the following lines:

```
#
# Batch Mail - batch mailing of a file to a LOT of users
#
# Usage: batch-mail "from" "subject" filename

sender_copy = $LOGIN
replyto = "The-Mr-Big-list"

fastmail -b $sender_copy -r $replyto -f "$1" -s "$2" $3 person1
sleep 10
fastmail -r $replyto -f "$1" -s "$2" $3 person2
sleep 10
fastmail -r $replyto -f "$1" -s "$2" $3 person3
sleep 10
fastmail -r $replyto -f "$1" -s "$2" $3 person4

< etc >
```

with the invocation:

```
batch-mail "Mr. Big" "Warning to all" warning.text
```

would mail a copy of the ’warning.text’ file to person1, person2, person3, etc. "$LOGIN" will also receive a copy of the first message in the mail, *silently*. Each resultant message will include the headers:

```
From: big-vax!big (Mr. Big)
Subject: Warning to all
Reply-To: The-Mr-Big-list
```

This program should turn out to be considerably faster than the alternative methods of accomplishing this task.

FILES

/usr/lib/sendmail	sendmail transport if available
/bin/rmail	transport if no sendmail
/tmp/fastmail.$$	temporary file

AUTHOR

Elm Development Group

SEE ALSO

sendmail(1), rmail(1), elm(1L)

BUG REPORTS TO

Syd Weinstein elm@DSI.COM (dsinc!elm)

COPYRIGHTS

NAME

filter - filter incoming messages before adding to mailbox

SYNOPSIS

filter [-q] [-v] [-l] [-o output] [-f rules-file]
filter [-n] [-f rules-file]
filter [-r] [-o output] [-f rules-file]
filter [-c] [-s] [-o output] [-f rules-file]
filter [-c] [-S] [-o output] [-f rules-file]

DESCRIPTION

Filter uses a set of selection rules to process incoming mail messages. The format for the rules, located in $HOME/.elm/filter-rules, is;

rule ::= **if** **(***expression***)** **then** *action*

where *expression* is:

expression ::= { **not** } *condition expr2*

expr2 ::= *NULL* | **and** *condition expr2*

condition ::= *field relationship value*
::= **lines** *relop numvalue*
or ::= **always**

These further break down as:

field ::= **from** | **subject** | **to** | **lines** | **sender**

relationship ::= **=** | **!=**

relop ::= **=** | **!=** | < | > | <= | >=

value ::= *any quoted string*

numvalue ::= *any integer value*

Action can be any of:

delete
save *foldername*
savecopy *foldername*
execute *command*
executec *command*
forward *address*
leave

For further information about the rules language, please see *The Elm Filter Guide*.

The flags the *filter* program understands are;

-c	Clear logs. If this flag is used, the log files will be removed after being summarized by either "-s" or "-S" (see below).
-f *rules-file*	Get rules from specified file instead of default $HOME/.elm/filter-rules

-l	Log actions only. Do not log information about mailed messages.
-n	Not really. Output what would happen if given message from standard input, but don't actually do anything with it.
-o *file*	Redirect stdout messages to the specified file or device. Logging of filter actions is not affected. (This is added to more gracefully deal with changes in the file system (since sendmail is quite picky about modes of files being redirected into.))
-q	Quiet. Do not log filtering actions.
-r	Rules. List the rules currently being used.
-s	Summarize. List a summary of the message filtering log.
-S	This is the same as '-s' but includes message-by-message information additionally.
-v	Verbose. Give output on standard out for each message filtered. Useful to have redirected to ''/dev/console'' and such. (see **-o** too).

AUTHOR

Elm Development Group

FILES

$HOME/.elm/filter-rules	Default rules file
$HOME/.elm/filterlog	A log of what has been done
$HOME/.elm/filtersum	A summary of what has been done
/etc/passwd	Used to get users home directory

SEE ALSO

The Elm Filter Guide
readmsg(1L), elm(1L), mail(1), mailx(1), sendmail(1,8)

COMMENTS

Not really smart about pattern matching.

BUG REPORTS TO

Syd Weinstein elm@DSI.COM (dsinc!elm)

COPYRIGHTS

NAME

frm,nfrm – list from and subject of selected messages in mailbox or folder

SYNOPSIS

frm [**–hnQqStv**] [**–s** *status*] [folder | username] ...
nfrm [**–hnQqStv**] [**–s** *status*] [folder | username] ...

DESCRIPTION

Frm outputs one line per message of the form:

from [subject]

where *from* is the name of the person the message is from, and *subject* is the subject of the message, if present. If the message is from you, the *from* portion will read ‘‘To user’’, where ‘user’ is the user the message was sent to. This happens when you receive a copy of a letter you sent.

If a folder is specified, the program reads that folder rather than the default mailbox. If the argument is a username then *frm* looks in that user’s mailbox, provided you have permission to read it.

A folder can be specified with the same notation as when invoking the **Elm** mailer (e.g., =folder).

Frm invoked as *nfrm* is identical to invoking ‘‘frm -s new’’.

OPTIONS

The *frm* program has the following options:

–h Print a brief help message summarizing the options.

–n Number the messages using the same numbering scheme that, for example, *readmsg* will understand.

–Q Very quiet mode. Only error messages are produced. This option is useful in shell scripts, where only the success or failure of the program is important, and output is not desired.

–q Quiet mode. Output only a one-line summary for each mailbox or folder specified.

–S Summarize the number of messages by message status in each mailbox or folder. If you want just a summary line, use this in conjunction with the **–q** option.

–s *status*
Only display headers from messages with the given status. ‘status’ is one of "new", "unread", "old" (same as "unread"), or "read". The **–s** option can be specified multiple times to print header information from, for example, only new and unread messages. It is sufficient to specify only the first letter of the status.

–t Tidy mode. If the *from* field is long enough to displace the subject field from its natural start column, move the subject down onto the next line.

–v Verbose mode. Print a descriptive header before listing the contents of each mailbox or folder.

EXIT STATUS

Frm returns a zero status ("true") if messages matching ‘status’ are present. *Frm* returns 1 if no messages matching ‘status’ are present, but there are some messages, returns 2 if there are no messages at all, or returns 3 if an error occurred. If multiple mailboxes or folders are specified, the exit status only applies to the last one examined. This can be used in scripts to determine what kind of mail a user has.

EXAMPLES

Some example uses:

```
$ frm
```

will display header information from all the messages in your mailbox, or ‘‘You have no mail.’’ if there are no messages in your incoming mailbox.

```
$ frm -s new
```

will display header information from all new messages in your mailbox, or ‘‘You have no new mail.’’ Note the slightly different diagnostic.

```
$ frm -s new -s unread guest
```

assuming you have the proper file permissions to read guest’s mail, will print out header information from all new and unread messages in guest’s incoming mailbox. If there are no messages, *frm* will print ‘‘guest has no mail.’’

```
$ frm -q -S
```

will print only a one line summary of how many read, unread, and read messages are in your incoming mailbox. For example, ‘‘You have 2 new messages, 3 unread messages, 23 read messages.’’

AUTHOR

Elm Development Group

SEE ALSO

readmsg(1L), elm(1L), mail(1), mailx(1)

BUG REPORTS TO

Syd Weinstein elm@DSI.COM (dsinc!elm)

COPYRIGHTS

©Copyright 1988-1992 by The USENET Community Trust
Derived from Elm 2.0, © Copyright 1986, 1987 by Dave Taylor

NAME

listalias - list user and system aliases

SYNOPSIS

listalias [-s | -u] [regular-expression]

DESCRIPTION

Listalias outputs a line per alias in both the user and the system aliases. Each line is of the form;

```
   <alias>     <address>  (<comment>)
```

If an optional regular expression is used, just the aliases that match the specified expression are listed. If not, they are all listed.

Either way, the output is sorted.

OPTIONS

–s Display only the system aliases. **–s** and **–u** are mutually exclusive.

–u Display only the user aliases. **–s** and **–u** are mutually exclusive.

AUTHOR

Elm Development Group

SEE ALSO

checkalias(1L), elm(1L), elmalias(1L), newalias(1L)

BUGS

Since the program uses a pipe to egrep, how expressions are evaluated is a function of *that* program, not this one!

BUG REPORTS TO

Syd Weinstein elm@DSI.COM (dsinc!elm)

COPYRIGHTS

NAME

messages - quick count of messages in mailbox or folder

SYNOPSIS

messages
messages folder-name

DESCRIPTION

Messages counts the occurrences of "ˆFrom " in either the current incoming mailbox or the specified folder. This is, in fact, a simple little shell script!!

AUTHOR

Elm Development Group

SEE ALSO

from(1L), elm(1L)

BUGS

Don’t be foolish...

BUG REPORTS TO

Syd Weinstein elm@DSI.COM (dsinc!elm)

COPYRIGHTS

NAME

newalias - install new elm aliases for user and/or system

SYNOPSIS

newalias [-g]

DESCRIPTION

Newalias creates new hash and data files from a text file. If the program is invoked with the *-g* (global) flag, the program updates the system alias files. Otherwise, the program looks for a file called *$HOME/.elm/aliases.text* and, upon finding it, creates files *$HOME/.elm/aliases.dir*, *$HOME/.elm/aliases.pag* and *$HOME/.elm/aliases* for the *elm* program.

The format that the program expects is;

```
alias, alias, .. = comment = address
```

or

```
alias, alias, .. = comment = alias, alias, ...
```

The first form is for an individual user such as;

```
dave, taylor = Dave Taylor = veeger!hpcnou!dat
```

or optionally as;

```
dave, taylor = Taylor; Dave = veeger!hpcnou!dat
```

to allow the aliases to be properly sorted by last name.

The second is for defining a group alias such as;

```
gurus = Unix Gurus = alan, john, dave, mike, richard,
                     larry, t_richardson
```

Note that lines can be continued at will, blank lines are accepted without error, and that any line starting with ’#’ is considered a comment and is not processed.

If the ‘comment’, or ‘address’ fields need to contain ‘=’ characters, those characters must either be escaped using a backslash character or the address must be inside doublequotes (‘"’).

Aliases are case insensitive so dave and Dave are both the same alias.

Finally, aliases can contain other aliases, and/or groups;

```
unix = Unix people = gurus, taylor, jonboy
```

FILES

$HOME/.elm/aliases.text	alias source for user
$HOME/.elm/aliases.dir	alias dbz directory for user
$HOME/.elm/aliases.pag	alias dbz hash file for user
$HOME/.elm/aliases	alias data file for user
/usr/local/lib/aliases.text	alias source for system
/usr/local/lib/aliases.dir	alias dbz directory for system

/usr/local/lib/aliases.pag	alias data hash table for system
/usr/local/lib/aliases	alias data file for system

AUTHOR

Elm Development Group

SEE ALSO

elm(1L), elmalias(1L), checkalias(1L), mail(1), mailx(1)

DIAGNOSTICS

Newalias has a couple of descriptive error messages which don’t need to be detailed here. It is, however, worth noting that the *checkalias* program can be used to ensure that the aliases are in the proper order and are available for the *elm* system.

BUG REPORTS TO

Syd Weinstein elm@DSI.COM (dsinc!elm)

COPYRIGHTS

NAME

newmail,wnewmail - programs to asynchronously notify of new mail

SYNOPSIS

newmail [-d] [-i *interval***] [-w]** *file-spec* {*file-spec*...}
wnewmail [-d] [-i *interval***] [-w]** *file-spec* {*file-spec*...}

DESCRIPTION

Newmail is a program to allow monitoring of mailboxes in an intelligent fashion. It is based loosely on *biff(1)* and the version of *newmail* that was distributed with **Elm 1.7**. The basic operation is that the program will check the list of specified mailboxes each *interval* seconds and will list any new mail that has arrived in any of the mailboxes, indicating the sender name, and the subject of the message. Each entry displayed can be in a number of different formats depending on the mode of the program, the number of folders being monitored, and the status of the message. If you're running it as a *window* (e.g. ‘‘-w’’ or invoked as *wnewmail*) then the output will be similar to:

sender name - subject of message
Priority: *sender name - subject of message*

where <sender name> is either the name of the person sending it, if available (the ARPA 'From:' line) or some other brief indication of origin. If you are the sender, <sender name> will be replaced by "to <recipient name>". If there is no subject, the message "<no subject>" will appear on the screen. Folders are indicated by having the folder name appear first on the output line, as in:

folder: *sender name - subject of message*

If you're running *newmail* without the windows option, then the output is more suitable for popping up on an otherwise active screen, and will be formatted:

>> New mail from *sender name - subject of message*
>> Priority mail from *sender name - subject of message*

Again, with folder names indicated as a prefix. The flags available are:

-d This will turn on the debugging, verbose output mode. It is not recommended that you use this option unless you're interested in actually debugging the program.

-i *interval* This will change the frequency that the program checks the folders to the interval specified, in seconds. The default interval for the program is 60 seconds. *Note: if you change the interval to less than 10 seconds, the program will warn you that it isn't recommended.*

-w Use of the ‘‘-w’’ flag will simulate having the program run from within a window (e.g. the more succinct output format, and so on). Most likely, rather than using this option you should be simply invoking *wnewmail* instead. *File specs* are made up of two components, the *folder name* and the *prefix string*, the latter of which can always be omitted. The format is *foldername=prefixstring*, and you can specify folders by full name, by simply the name of the user whose mailbox should be monitored, or by the standard **Elm** metacharacters to specify your folder directory (e.g. ‘‘+’’, ‘‘=’’, or ‘‘%’’). Folders that cannot be opened due to permission mismatches will cause the program to immediately abort. On the other hand, files that do not exist will continue to be checked every *interval* seconds, so some care should be exercised when invoking the program. The program will run until you log out or explicitly kill it, and can internally reset itself if any of the folders shrink in size and then grow again. The default folder to monitor is always your incoming mailbox.

EXAMPLES

Some example invocations:

```
$ newmail
```

will check your incoming mailbox every 60 seconds.

```
$ newmail  -i  15  joe  root
```

will monitor the incoming mailboxes for ‘‘joe’’ and ‘‘root’’, checking every 15 seconds for new mail.

```
$ newmail  "mary=Mary"  +postmaster=POBOX
```

will monitor the incoming mailbox for user ‘‘mary’’, prefixing all messages with the string ‘‘Mary’’, and the folder in the users *maildir* directory called ‘‘postmaster’’, prefixing all of those messages with ‘‘POBOX’’. You can also have more complex monitoring too, for example:

```
$ newmail  -i  30  $LOGNAME=me  ${LOGNAME}su=myroot   /tmp/mbox
```

will check every 30 seconds for the users mailbox, a mailbox that is the users login-name with ‘‘su’’ appended (e.g. ‘‘joe’’ would become ‘‘joesu’’) and the file /tmp/mbox, prefixing new mail for each with ‘‘me’’, ‘‘myroot’’ and ‘‘mbox’’ respectively.

AUTHOR

Elm Development Group

SEE ALSO

notify in sh(1) or csh(1)

BUG REPORTS TO

Syd Weinstein elm@DSI.COM (dsinc!elm)

COPYRIGHTS

NAME

printmail - format mail in a readable fashion for printing

SYNOPSIS

printmail {-p} {filename}

DESCRIPTION

Printmail copies all messages from either the file specified or the default user mailbox with each message separated by a line of dashes. With the **–p** option, the messages are separated by a form feed instead.

In fact, this command is actually just a slightly differently formatted call to *readmsg*:

```
readmsg {-p} {-f filename} *
```

which does all the work here.

EXAMPLE

A typical usage of this command is;

```
printmail | lpr
```

AUTHOR

Elm Development Group

SEE ALSO

readmsg(1L)

BUG REPORTS TO

Syd Weinstein elm@DSI.COM (dsinc!elm) ©Copyright 1988-1992 by The USENET Community Trust

Derived from Elm 2.0, © Copyright 1986, 1987 by Dave Taylor

NAME

readmsg - extract messages from a mail folder

SYNOPSIS

readmsg [**-anhp**] [**-f** folder] [selection ...]

DESCRIPTION

Readmsg extracts selected mail messages from a mail folder. One helpful use of the program is when you are composing a response to a mail message in an external editor. In this case, you can run *readmsg* to pull a copy of the original message into the editing buffer. When you run *readmsg* from within *elm* (e.g. from a subshell escape or in an external editor while composing a mail reply) it behaves a bit differently from when you run it directly (e.g. from a shell command line). We will first describe its normal behavior, and then describe how it is different when you run it under *elm*. You tell *readmsg* which messages to extract with the *selection* argument. There are a couple of possible different ways to specify the *selection*.

1. A lone ‘‘*’’ means select all messages in the mailbox.
2. A list of message numbers may be specified. Values of ‘‘0’’ and ‘‘$’’ in the list both mean the last message in the mailbox. For example:

   ```
   readmsg 1 3 0
   ```

   extracts three messages from the folder: the first, the third, and the last.
3. Finally, the *selection* may be some text to match. This will select a mail message which *exactly* matches the specified text. For example,

   ```
   readmsg staff meeting
   ```

   extracts the message which contains the words ‘‘staff meeting.’’ Note that it will *not* match a message containing ‘‘Staff Meeting’’ the matching is case sensitive. Normally only the first message which matches the pattern will be printed. The **–a** option discussed in a moment changes this. The **-f** flag indicates that you’d rather use the folder specified rather than the default incoming mailbox. The specified *folder* can be a filename or a specification such as ‘‘=sentmail’’. The **-h** flag instructs the program to include the entire header of the matched message or messages when displaying their text. (default is to display the From: Date: and Subject: lines only) The **-n** flag instructs the program to exclude *all* headers. This is used mostly for extracting files mailed and such. The **-p** flag indicates that the program should put form-feeds (control-L) between message headers. The **-a** flag indicates that all messages which match the pattern specified on the command line should be printed, not just the first. If a pattern was not specified on the command line then this flag has no effect. When you run *readmsg* under *elm* (once again, say in the context of an external editor) the behavior will be different from that described above as follows.

1. The default mail folder will be the folder you are currently examining in *elm* and not necessarily your incoming mail folder.
2. You do not need to specify a *selection* on the command line. If you omit the *selection* then *readmsg* will extract the message(s) you have selected in Elm. If you have tagged any messages then this would be all of the tagged messages, otherwise it would be the message you are currently examining.
3. Normally the message numbers *readmsg* uses are in mailbox order. When you call *readmsg* under *elm* and do not override the folder selection with the **–f** option, then message numbers will be sorted as they are displayed on the *elm* message index screen.

EXAMPLES

First off, to use this from within **vi** to include the text of the current message, you could use the command:

```
:r !readmsg
```

(as you hit the ':' the editor will put you at the bottom of the screen with the ':' prompt). The space following ':r' is required.

Let's look at something more interesting, however;

Suppose you have the mail file;

```
From joe Jun 3 1986 4:45:30 MST
Subject: hello

Hey Guy!  Wanta go out and have a milk this evening?

Joe

From john Jun 3 1986 4:48:20 MST
Subject: Dinner at Eight
From: John Dinley <xyz!john>

Remember you should show up about eight, okay?

                              - John D -

From xxzyz!cron Jun 3 1986 5:02:43 MST

Cannot connect to server: blob
Job 43243 deleted from queue.
```

The following commands will result in;

```
$ readmsg 2
[ display the second message, from John ]

$ readmsg
[ an error, unless we're calling from elm ]

$ readmsg BLOB
[ no match - case sensitive! ]

$ readmsg -h connect to server
[ displays third message, including headers ]
```

FILES

/usr/mail/<username>	The incoming mail
$ELMSTATE	Status information from *elm*

AUTHOR

Elm Development Group

SEE ALSO

newmail(1L), elm(1L)

BUGS

The '∗' metacharacter doesn't always work as expected!
Perhaps the pattern matching should be case insensitive?

It might be confusing that messages are sorted when running under *elm* with the current folder, and in mailbox order for all other cases.

BUG REPORTS TO

Syd Weinstein elm@DSI.COM (dsinc!elm)

COPYRIGHTS

NAME

wnewmail - daemon to asynchronously notify of new mail

SYNOPSIS

wnewmail
wnewmail filename

DESCRIPTION

Wnewmail is a daemon designed to run in **a window** on a windowing system (such as an HP or Sun system) and check every 10 seconds to see if there is any new mail for the user that started it up. If there is new mail, the program will "beep", and write to the window for each of the new messages;

Mail from <name> -- <subject>

where <name> is either the name of the person sending it, if available (the ARPA 'From:' line) or machine!login where machine is the machine the mail was sent from. If there is no subject, the message "<no subject>" will appear on the screen. If the message is a *priority* message (that is, the header contains a line "Priority:"), then the line output will be "PRIORITY mail from ..." rather than just "Mail from". This program will run forever, and can internally reset itself if mail is deleted from the incoming mailbox while trying to monitor it. If **wnewmail** is started up with a filename, it will perform exactly the same, but with the specified file as the one to check rather than the default users mailbox.

AUTHOR

Elm Development Group

SEE ALSO

notify in sh(1) or csh(1), newmail(1L)

NOTE

This is almost identical to the program **newmail**...

BUG REPORTS TO

Syd Weinstein elm@DSI.COM (dsinc!elm)

COPYRIGHTS