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 **–g** 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.