Lafite
The Interlisp Mail System

Documentation: {Eris}<Lisp>Library>Lafite.Press
Program: {Eris}<Lisp>Library>Lafite.Dcom
Revised: February 22, 1984 by Bill van Melle

General comments

Lafite is the Interlisp system for reading and sending mail. Lafite retrieves inbound mail from the user's *inboxes* on one or more mail servers. Mail is retrieved into one or more *mail folders*, which can be any Interlisp-accessible file. One can open a *browser* on a mail folder, which allows the folder's contents to be displayed, deleted, moved into other folders, hardcopied, etc. Messages can be composed using the standard Interlisp text editor and all its facilities, then sent to other users.

Lafite permits communication with various sorts of mail servers. Currently implemented are interfaces to the Grapevine mail system and the NS mail system.

Mail systems are notorious for inspiring lengthy wish lists for new functionality or user interface embellishments. Feel free to send your suggestions to LafiteSupport (using Lafite's ‘‘Lafite Report’’ form, described below).

Compatibility with Laurel

Previous users of the Laurel mail program will find Lafite to have a similar user interface. Some of the ways in which it differs from Laurel are the following:

> Laurel can only access mail folders on the local disk; Lafite can access folders on remote file servers. Thus, it is not necessary to transfer mail folders back and forth between your local disk and your file servers.
>
> Laurel can only ‘‘browse’’ one mail folder at a time; Lafite can have several mail folders ‘‘opened’’ at the same time. Utilizing the Interlisp window system, you can view the table of contents of many mail folders and refer to messages in them independently.
>
> You may have multiple windows displaying messages and multiple windows for sending messages and you may move text freely among them.

Lafite can read mail files written by the Laurel and Hardy mail programs; the files it writes are in Laurel format.

Loading Lafite

Lafite consists of two parts: a generic part that manages browsers, mail folders and message composition; and a protocol-specific part that manages communication between Lafite and the remote mail system. You have to load both parts. The first is the file `LAFITE.DCOM`, plus the files that it automatically loads. The second depends on your network environment. For sites that use the Xerox NS Mail system, you should load the file `NSMAIL.DCOM`. For Grapevine sites, load the file `MAILCLIENT.DCOM`. If your site has a choice of mail server types, you can load all the relevant protocol files and then choose among them by setting the mode (see **Lafite Modes**, below).

Lafite operation

When Lafite is loaded, it can be started by calling `(LAFITE 'ON` *MAILFOLDER*`)`. Lafite will attempt to establish the user's mail server identity, read stored user data from the file `LAFITE.PROFILE` (if it exists) and bring up the Lafite Status window. It will then establish a browser for the file *MAILFOLDER*, creating an empty mail folder of that name if one does not exist. If *MAILFOLDER* is not supplied then `ACTIVE.MAIL` will be used (see

DEFAULTMAILFOLDERNAME below). If *MAILFOLDER* is supplied but is the atom NIL, then no mail folder is opened but you are free to send mail and open any mail folder at a later time.

Mail files must reside on randomly-accessible devices. In the current Interlisp environment, this means they must reside on {DSK}, {FLOPPY}, or a file server that supports the Leaf protocol.

There are three major types of windows used by Lafite: the Status Window; Browser Windows, which are views on particular mail folders; and Message Composition Windows. Each type of window has its own fixed menu of commands. In general, while a command is ‘‘in progress’’, the menu item that invoked it is greyed out. Commands may be selected with the LEFT or MIDDLE mouse buttons; in some cases, the MIDDLE button provides some sort of special treatment, described in the documentation of such commands.

The Lafite Status Window

The Lafite Status window contains a small, fixed menu and a region for Lafite status information. While Lafite is in operation, it runs a background process—LAFITEMAILWATCH—that polls the user’s inboxes periodically and reports in the status region if there is new mail. Clicking in the status region causes the background process to wake up and report status immediately, instead of waiting its normal interval. The commands in this window’s menu are as follows:

Browse — pops up a menu of the mail folders that Lafite is aware that you have. Selecting ‘Another Folder’ prompts you for a folder name in the prompt window. After a folder is selected, a *browser* window onto that mail folder is opened.

Selecting the **Browse** command with the MIDDLE button brings up a menu of Browse-related commands. In addition to simple **Browse** there are these commands:

Browse Laurel Folder — Browses a file that was produced by the Laurel mail reader version 6.1 or later. Laurel 6.1 files are almost the same as Lafite files, but contain some line-formatting information that is stripped out by this command. After you have applied this command once to a file, you can subsequently browse the file with the normal **Browse** command (unless you use Laurel on it again, of course). **Important Note**: this command is not intended for repeated use, but simply to make mail files built by Laurel more pleasing to browse. This command has the side effect of destroying any Tedit formatted messages, so should be used with care.

Forget Folder — Removes a folder from the list of known mail folders.

Forget Message Form — Removes a message form from the list of known message forms (see **Save Form** command).

Send Mail — brings up an Interlisp text editor window on a message form. The form is a canonical ‘‘empty message’’ if **Send Mail** is selected with the LEFT button. If **Send Mail** is selected with the MIDDLE button, a menu is presented with the following choices:

Standard Form—provides an empty message template (same as using the LEFT button).

Lisp Report—provides a message template to report an Interlisp bug or make a suggestion.

Lafite Report—provides a message template similar to **Lisp Report** but sent to Lafite maintainers.

Saved Form—prompts for a form name, which can be any text file, or a form created by the **Save Form** command (below).

Also in the menu are any names of known user-defined message forms created by the **Save Form** command. Each message form runs in its own process, so you can have several in progress at once. When you have finished composing the message, click **Deliver** in the message’s menu.

Quit — Stops Lafite and closes all browser windows. If any of the associated mail folders need updating, prompts with a menu asking what degree of updating should be performed (see **Update** command). It also saves the names of known mail folders and form files on the file `LAFITE.PROFILE` so that this information will be available when you next run Lafite. You may achieve the same result under program control by calling `(LAFITE 'OFF)`, rather than buttoning this menu item. Most users find that they never invoke ‘Quit’, but rather keep Lafite always active in the background.

Selecting the **Quit** command with the MIDDLE button brings up a menu of status changing commands. This menu includes items for changing Lafite’s mode (see **Lafite Mode**), and the command **Restart**, which is equivalent to `(LAFITE 'OFF)` followed by `(LAFITE 'ON NIL)`.

Browser Windows

A Browser window is a view onto a mail folder. The main part of the window displays the table of contents, a one-line summary of each message. This window is scrollable in both dimensions. Above the table of contents is a horizontal menu containing commands specific to the mail folder associated with the browser window. Above the menu is a prompt window, in which various status information related to the browser is printed, and where some information is prompted for.

Browser comands operate on the currently selected set of messages. A selected message is indicated by a black triangle to the left of its message number. A message can be selected by clicking anywhere inside its summary line. The type of selection depends on which button is used:

left button — selects just this single message, deselecting any other selected message.

middle button — adds a message to the current selection.

right button — extends the selection up or down. Deleted messages are not included in this extension unless the control key (CTRL) is down.

shift key (SHIFT) and any button — removes this message from the current selection.

Laurel users note: messages can be selected by clicking *anywhere* within the summary line (except for the mark area, see below), unlike in Laurel, where you must select at the left end.

The commands in the browser menu are as follows:

Display — displays the selected message. If the selected message is already displayed, the selection is advanced to the next undeleted message, and this message is displayed. If there is more than one message selected, buttoning **Display** cycles through the messages.

If you select **Display** with the LEFT button, the message is displayed in the primary message display window for the browser, replacing any previously displayed message. If you select **Display** with the MIDDLE button, the message is displayed in a newly created window, for which you will be prompted. Using the MIDDLE button you can make multiple windows containing messages for further reference (e.g., to use in composing your own message).

Delete — deletes the selected messages. A deleted message is indicated by a black line through its summary line. The message is not actually removed from the mail folder until you Expunge (see Update, below).

Undelete — undeletes the selected messages.

Answer — constructs a Message Composition Window containing an *answer template* for the current message. After you deliver the answer, an ‘‘**a**’’ will appear in the browser window as the message’s mark.

Forward — similar to ’Answer’ but the message form is a *forward template* for the currently selected messages. After you deliver the forwarded the messages, an ‘‘**f**’’ will appear as the message’s mark.

Hardcopy — prints the selected messages on your local printing device. When the hardcopy is complete, the message’s mark is changed to ‘‘**h**’’ if the message didn’t already have a more interesting mark. Messages can be marked for hardcopy, but the actual printing deferred until later; see description of `LAFITEHARDCOPYBATCHFLG`. Currently, one should exercise care in hardcopying large sets of messages, as Lafite makes no attempt to perform the hardcopying in smaller pieces; too large a body of messages can make printers unhappy, or exhaust local disk space in the process.

Move To — pops up a menu of known mail folders and moves the selected messages to the chosen folder. A new mail folder can be created by selecting ‘Another Folder’ and typing in the mail folder name in the prompt window. You will then be asked to confirm the move. When the move is successful, the messages are marked deleted in the source browser window, and given the ‘‘**m**’’ mark.

The name of the mail folder you most recently moved messages to appears in the title bar of the browser window as the ‘‘Default ’Move To’:’’ folder. You can ‘‘accelerate’’ subsequent Move operations by selecting the ’Move To’ command with the MIDDLE button. This will perform the move to the Default ’Move To’ folder without bringing up a menu.

Update — The changes that you make to your mail folder (deletions, changes of message marks, etc) are not actually transmitted to the physical mail file until you perform the Update command. There are two subcommand choices for Update:

> **Write out Changes Only** — makes the browser and the mail file completely consistent with one another: if you were at this point to logout from Lisp, run Lafite in another incarnation of Lisp, and browse the same mail folder, you would get a browser that was in exactly the same state, deleted messages and all. This command involves writing out to your mail file and its table of contents the information about what has changed: new marks, deletions, newly parsed messages.
>
> **Expunge Deleted Messages** — in addition to making the browser and the mail file completely consistent with one another, this command compacts your physical mail file so as to remove all messages marked deleted.

Which to choose? You eventually want to Expunge, so as to reduce the amount of file space your mail requires, but Expunge is not always fast. **Expunge** requires rewriting your mail file starting at the first deleted message, so is somehow ‘‘proportional’’ to the number of undeleted messages beyond that point. **Write out Changes Only** is proportional to the number of changes, and is thus usually faster than **Expunge**, except when the changes consist primarily of a string of deleted messages at the end of the folder.

If you Close or Shrink a Browser window that has had changes to it, you are prompted with a menu offering to **Write out Changes**, **Expunge**, or make no change before the window is closed/shrunk.

If you have deferred hardcopy, the Update menu also includes a choice **Do Hardcopy Only** if you want to print out your messages but not actually update the file.

Get Mail — brings new mail into the folder that this browser is viewing.

Changing the Message Mark

Each message in a mail folder has a ‘‘message mark’’, which is an additional tidbit of information about the message displayed in the summary line in the browser. Messages originally come in with the mark ‘‘**?**’’, meaning they are unexamined. Some Lafite commands change the mark automatically. For example, displaying a message changes its mark from ‘‘**?**’’ to a blank; answering a message changes its mark to ‘‘**a**’’. You can change the mark directly by selecting with the mouse in the narrow area immediately to the left of the message number, where the mark is printed. Simply click in the mark position, and type the new mark (a single character).

The Table of Contents

To speed the browse operation, Lafite maintains, for each mail folder you browse, a ‘‘table of contents’’ file, which contains all the information displayed in the browser window. This file is named by concatenating the name of the mail file with ‘‘`-LAFITE-TOC`’’. This file is completely redundant, in that if it doesn’t exist, Lafite simply recomputes it by parsing the entire mail file. Lafite has some rudimentary checks to guarantee the consistency of the contents file with the mail file it describes; Lafite deletes and recomputes the contents if it suspects that the table of contents file is out of date.

Message Composition Windows

On top of the text editor window is a horizontal menu for telling Lafite what to do with the message being created in the text editor window. When you have transformed the text to the desired message, select one of the following menu items:

Deliver — sends your message. This process happens in background, so you can proceed with anything else you desire while delivery proceeds. If successful, the window closes and an entry is made in your *outbox* (see below). If the delivery fails, for any of a variety of reasons (e.g. bad address fields, the mail server timed out) which will be displayed in the prompt window, the message is redisplayed and you are back in the text editor to re-edit and then resend the message. During the delivery process, the menu atop the message window changes into a single item, **Abort**; if you click this item, the delivery is aborted, and you are returned to editing your message.

Save Form — asks you for a file name on which to save this message for later use. This is the way to save a message form for later repetitive use. It does *not* send it. If no extension is given for the file name, it defaults to the value of `LAFITEFORM.EXT` (initially `LAFITE-FORM`).

Of course, you can always bring up the text editor command menu by MIDDLE buttoning the title bar of the editor window. You can then do any text editor command. If you select ‘Quit’ you will immmediately leave the text editor, bypassing the above commands. You can also simply close a window using the normal window command menu if you decide you don’t want to do anything with a message you started composing.

When editing message forms, fields that should be filled in are enclosed in ‘‘>>’’ and ‘‘<<’’; e.g., >>Recipients<<, >>Subject<<. To make it easier to fill these in, the first field is highlighted in delete mode. Each successive field can be reached by typing the middle-blank key on the keyboard (or OPEN on the Dandelion keyboard, or whichever key you have assigned the TEdit ‘‘Next’’ syntax to). See the TEdit documentation for details.

In a saved form, if you insert a field ‘‘>>Self<<’’, it will be automatically filled with the name of the currently logged in user when retrieved into a message composition window. This facilitates making forms that many different people can use. Of course, if you edit an already saved form containing ‘‘>>Self<<’’ in a message composition window, you will have to perform the inverse, replacing your name with ‘‘>>Self<<’’, before saving it away again.

After a message is succesfully delivered, it is entered into your *outbox*, a window attached to the bottom of your status window. This window contains a one line description of each of the most recent *n* messages you have sent, where *n* is the value of the variable `LAFITEOUTBOXSIZE`, initially 2. The outbox is treated as a menu—selecting a line in it brings up the corresponding message for further editing and delivery. The outbox can be independently closed, if you are no longer interested in the messages displayed therein and want to free up the resources that they are tying down.

Format of Message Headers

The header of a message contains a number of fields, each on a separate line, followed by a blank line to separate the header from the message body. Each header line consists of a field name followed by a colon, then the contents of the field. Messages standardly have ‘‘To:’’ and ‘‘cc:’’ fields consisting of one or more recipient names, separated by commas (and spaces if desired). Lafite automatically supplies a ‘‘From’’ field containing your name. If you want the message to appear to be ‘‘from’’ someone else (e.g., if you are sending the message from someone else’s logged in Lafite), or from more than one user, you can supply your own ‘‘From’’ field. In this case, Lafite will supply a ‘‘Sender’’ field to show who actually sent the message.

Sending Formatted Messages

You have available to you the full power of TEdit when you are composing a message. This means you can change fonts, format paragraphs, and insert ‘‘image objects’’. If you try to deliver such a formatted message, Lafite will ask if you want to retain the formatting information, putting up a menu with these choices:

Send Formatted Message — retains all the formatting information. Note that only Lafite users can read formatted messages; all other mail readers will see the plain text of the message. Even if all of the recipients are Lafite users, keep in mind that retaining the formatting information means in particular retaining the fonts you used in composing the message; not everyone likes to read mail in the font you chose, so if the only ‘‘formatting’’ is a nonstandard font that isn’t important to the appearance of the message, do not make this choice.

This choice is made automatically if the message contains image objects, as there is no way to send the images without the formatting.

Send Plain Text — sends only the text of the message. The message will appear to the recipient in whatever font her mail reader standardly uses, and all paragraph formatting (centering, justification, special tabstops) will vanish.

Abort — does not send the message, but returns to the message editor to allow you to continue editing the message.

Lafite Modes

Lafite is capable of sending and retrieving mail via different protocols. At any one time, however, it only operates with a single protocol set, which is considered its current ‘‘mode’’. The currently implemented modes are GV (Grapevine) and NS. The easiest way to change the mode is to use the MIDDLE-button menu under **Quit**. Lafite’s mode can also be changed programmatically with the function `LAFITEMODE`:

`(LAFITEMODE` *MODE*`)`

Returns the current mode, a litatom. In addition, if *MODE* is non-`NIL`, changes Lafite’s mode to be *MODE*.

When Lafite is first turned on, it chooses its mode as follows: If there is only one mode supported (i.e., only the files supporting one protocol have been loaded), it chooses that mode; otherwise, if `LAFITEMODEDEFAULT` is non-`NIL`, it chooses that mode; otherwise, the user must set the mode manually. The current mode is displayed in the status window if `LAFITESHOWMODEFLG` is true and more than one set of mail protocol implementations has been loaded.

You can freely intermix mail of the various modes in one folder, but the current implementation is not very clever about it. For example, the Answer command always treats the selected message as if it were one in the current mode. So if you try to answer a Grapevine message while in NS mode, some confusion may result.

Grapevine

The Grapevine implementation of mail protocols is obtained by loading the library file `MAILCLIENT.DCOM`.

Grapevine addresses are of the form ‘‘name.registry’’, e.g., ‘‘Carstairs.pa’’. If you omit the registry, it is defaulted to the value of `DEFAULTREGISTRY`. Arpanet recipients are of the form ‘‘name@host’’. In addition, the following forms of address are recognized, where ‘‘actual address’’ must be a valid Grapevine or Arpanet address:

Human sensible name <actual address>

actual address (random comments)

"Comments, including parentheses and commas" <actual address>

If you address a message to a public distribution list (a Grapevine name ending in the character ‘‘^’’) and have not included a Reply-to field, Lafite will prompt you to supply one. Your choices are to include a Reply-to: self field, so that answers to this message will be sent only to you; a Reply-to: other field (you get to fill it in yourself), or no Reply-to: field at all. It is important to have a Reply-to field in messages sent to distribution lists, so that casual users will not inadvertantly reply to the entire distribution list when a reply to you only was intended.

Please be careful in using public distribution lists. Keep in mind that your one message will be received by many people—is what you have to say important enough to be worth taking their time? There are certain organizational distribution lists, e.g., AllPA^.pa, or ISL^.pa, that you should definitely avoid sending casual messages to, as they are large and their members are not voluntary. Other distribution lists are really ‘‘interest lists’’ whose members have voluntarily consented to receive messages in the general category of movies, concert announcements, humorous anecdotes, or whatever. Messages to these lists are less constrained, but you should still take a moment to think about whether your message is appropriate. I highly recommend that all electronic mail users read Chapter 6 of the Laurel manual, ‘‘Message system mores’’.

Shortcomings as of this writing: Private distribution lists are not yet supported. And the Answer command does not preserve the entire text of the variant forms listed above, only the ‘‘actual address’’.

NS Mail

The NS implementation of mail protocols is obtained by loading the library file `NSMAIL.DCOM`.

NS addresses are of the form ‘‘name:domain:organization’’, e.g., ‘‘Carstairs:PARC:Xerox’’. If you omit the domain and/or organization, they are defaulted to your own domain and organization. NS mail does not support the many variant forms of address that Grapevine does, because the header is not actually sent as text, but as a structured set of formally named recipients.

Some users of the NS Mail system send messages in a format that Lisp does not understand. The current Lafite implementation leaves such messages in your inbox to be retrieved by other means; all that you see back from GetMail is a header and a note that there was an attachment that Lafite couldn’t read.

Changing the Lafite User—Lafite customization

`(LAFITE` *ON/OFF MAILFOLDER* . *OPTIONS*`)`

Used for starting and stopping Lafite with an optional mail folder. If *ON/OFF* is the atom `ON` then Lafite will start processing using *MAILFOLDER*. If *MAILFOLDER* is not supplied then Lafite will use your `ACTIVE.MAIL` mail folder (actually, the value of `DEFAULTMAILFOLDERNAME`). If *MAILFOLDER* is supplied but is the atom `NIL` then no browser will be created but you can send messages and start browsing mail files later using the ’Browse’ menu item.

If *ON/OFF* is the atom `OFF` then Lafite will close all mail folders, expunging all messages marked for deletion; close all windows associated with Lafite; and remove the mail watching process from the active process list. This is the same as invoking ’Quit’ in the Lafite Status Window.

The third and subsequent arguments are options. The only one currently recognized is the atom `SHRINK`, which instructs Lafite to immediately shrink the browser window brought up on *MAILFOLDER*. In this case, if *MAILFOLDER* is (explicitly) `NIL`, it still defaults to `DEFAULTMAILFOLDERNAME`, since `SHRINK` makes no sense otherwise.

Lafite uses its own host and directory names for mail folders, `LAFITE.PROFILE`, etc., rather than the current connected directory because you may want to keep your mail folders someplace special (e.g., the local disk or your login directory), and the connected directory can change. The global variable `LAFITEDEFAULTHOST&DIR` is provided to tell Lafite where you generally keep your mail. `LAFITEDEFAULTHOST&DIR` should be atomic, in the same form as `LOGINHOST/DIR` (e.g. `{PHYLUM}<CARSTAIRS>`). If `LAFITEDEFAULTHOST&DIR is NIL`, then the value of `LOGINHOST/DIR` is used—i.e., your login host and directory.

If you are running in a Lisp in which some previous user has been using Lafite, you need to take some action to get Lafite to work on your mail files and in your name. When you change your user identity by calling `(LOGIN)` to log in as yourself, Lafite notices that the current user has changed, and attempts to authenticate the new user. However, Lafite still doesn't know how you want your Lafite customized; in particular, Lafite notices the value of `LAFITEDEFAULTHOST&DIR`, and loads your profile of known folders only when Lafite is first started. Thus to change the identity of the Lafite user, you should follow the following procedure:

1) Turn Lafite off, by clicking Quit in the status window, or calling `(LAFITE 'OFF)`.

2) Log in as the new user by calling `(LOGIN)`.

3) Set `LAFITEDEFAULTHOST&DIR` and/or `LOGINHOST/DIR` as appropriate, and any other personal variables that affect Lafite that matter to you. A typical way to do this step is to call `(GREET)` to get your personal initialization loaded.

4) Restart Lafite by calling `(LAFITE 'ON)`.

When things go wrong...

Lafite tries fairly hard not to break, but it is running in a very open environment, where users are free to interrupt arbitrary processes, network servers come and go, etc. As of this writing, the following are some abnormal states of note:

Mail file does not parse. This shows up when you browse a folder, Lafite starts parsing the folder, and encounters an error in the file, typically an incorrect message length. Lafite prints a message to this effect in the browser window and aborts the browse. The information Lafite prints includes the header of the last message parsed, and a byte pointer where the problem was encountered. The byte pointer is such that if you truncated the file to that pointer position, the mail file would be valid.

Solution: scavenge the mail file. The Library package `MAILSCAVENGE` contains a mail file scavenger. The Laurel MailFileScavenger program also works on Lafite files, as does the Hardy scavenger (I think).

The most common way to get a mail file into an inconsistent state is to abort a MoveTo or Update command somewhere in the middle (either manually, or because a remote server crashed). In the case of MoveTo, the problem is usually that the last message in the file is ‘‘too short’’, since it never was completely written. If the first operation you perform on the destination file after the crash is to browse it, Lafite will (usually) detect this situation and let you browse the file anyway, with a warning that the last message is truncated. Updating the file will then correct the length of the last message. Thus, in this case, you will not need a scavenger. If you neglect to browse the destination file before moving additional messages to it, however, you will need to scavenge.

Table of contents inconsistent with mail file. In theory, this should never happen; however, practice shows that it does. Lafite breaks with a message to this effect when it tries to operate on a message that isn't where it thought it was in the file. The appropriate action to take is to close the browser, selecting **Don't Update**, delete the table of contents file (the file with `-LAFITE-TOC` appended to the name), and then browse the file again. If this is not successful, you may need to scavenge the file. If you had made many changes to the browser (deletions, for example) that you would rather not lose, you can try selecting **Write out Changes Only** when you close the browser; this may succeed if the inconsistencies in the table of contents did not intersect with your changes.

Slow file server. If your mail files live on a remote file server that is particularly unresponsive, it may happen that the mail server connection over which new mail is being retrieved times out before the file server acknowledges receipt of the messages. The usual consequence of this is that your inbox is not flushed, so your new mail is in two places: your inbox, awaiting retrieval, and your mail file, to which it was just retrieved. A less common occurrence is that the mail server times out partway through the retrieval process, resulting in a Lisp break. You can ^ out of the break to return to the state before the GetMail started.

If this is often a problem for you, you may want to adopt the following idiom, which several people have found useful in maintaining the flexibility of remote mail files while utilizing the speed and reliability of the local disk. Keep most of your mail files on the remote server, as usual, but keep your ‘‘active’’ mail file, the one to which you standardly retrieve mail, on your local disk. Retrieve mail to this file, and dispatch from there to your remote files (using MoveTo) some or all of the messages you wish to keep. Mail files on {DSK} have very predictable performance during GetMail, which is good for both you and the mail server. Files on {DSK} are also less subject to other vagaries of remote servers (e.g., sudden crashes) that sometimes cause problems with mail files. And if you tend to delete much of your incoming mail after reading it once, you will find it much faster to keep your active mail on {DSK}, even if your remote server isn’t flaky.

You can also choose to keep most or all of your mail files on {DSK}, backing them up to a file server periodically. The LispUsers package COPYFILES is helpful for doing the backup automatically.

Lafite Profile Variables

Below are the global variables that control Lafite’s behavior.

DEFAULTREGISTRY

The name of your local registry, used if your login name does not include a registry. This is normally set in the local site-specific INIT.LISP file.

LAFITEDEFAULTHOST&DIR

The directory on which Lafite looks for LAFITE.PROFILE and all mail folders and message forms you access if you don’t supply an explicit directory. See ‘‘**Changing the Lafite User**’’ above.

DEFAULTMAILFOLDERNAME

If no mail folder is supplied to the function LAFITE, i.e., you call (LAFITE 'ON), then the value of this variable is used. Initially, ACTIVE.MAIL.

LAFITEMAIL.EXT

The default extension for names of mail folders. Initially, MAIL.

LAFITETOC.EXT

The string appended to the name of a mail folder to produce the name of its table of contents file. Initially, -LAFITE-TOC.

LAFITEFORM.EXT

The default extension for names of user-defined form files. Initially, LAFITE-FORM.

LAFITEFORMDIRECTORIES

A search path for Lafite forms, initially NIL. When you choose the **Saved Form** command underneath **Send Mail**, the form name that you enter is first searched for on your default directory (LAFITEDEFAULTHOST&DIR), and if not found there, Lafite searches the directories in the list LAFITEFORMDIRECTORIES. LAFITEFORMDIRECTORIES is typically set to a list of one or more public directories on which generally useful forms have been collected.

MAILWATCHWAITTIME

The number of minutes between polling for new mail from your mail servers. Initially set to 5.

LAFITEFLUSHMAILFLG

If NIL, Lafite won't flush your inbox when retrieving new mail, so the mail will still be there when you invoke Get Mail again. Initially, T.

LAFITENEWPAGEFLG

If T, then the Hardcopy command will start each message on a new page. Otherwise it will separate each message by a line of dashes. Intially, T.

LAFITEHARDCOPYBATCHFLG

If you often request hardcopy of single messages, one at a time, you may notice some disadvantages: short messages sometimes get lost in amongst other users long output, and they are wasteful of paper. And if you hardcopy large messages, you may not always care to wait around while the message is formatted for hardcopy. LAFITEHARDCOPYBATCHFLG is provided to allow you to postpone the hardcopying until it can be done all at once.

When this flag is true, Lafite ‘‘batches’’ your hardcopy requests, and doesn't actually print them until you do an Update, at which point it sends them all to the printer in one batch. When you have hardcopy pending, the Hardcopy button is speckled to remind you of this fact. The Update button has an additional choice **Do Hardcopy Only** in case you want to get your batched hardcopy printed without doing an actual Update.

The behavior of LAFITENEWPAGEFLG when batching hardcopy is that it applies only to the messages selected at each Hardcopy invocation; each new set of messages starts on a new page, independent of the setting of LAFITENEWPAGEFLG.

LAFITEHARDCOPYBATCHFLG is initially NIL.

LAFITEHARDCOPY.MIN.TOC

If non-NIL, is a positive number. Whenever Lafite is instructed to produce hardcopy for more than LAFITEHARDCOPY.MIN.TOC messages, it also produces a table of contents as a cover page for the hardcopy. Currently, this flag is only noticed if LAFITEHARDCOPYBATCHFLG is NIL. Initially NIL.

LAFITEDISPLAYAFTERDELETEFLG

If T, Lafite will display the next message if you delete the one that is in the message display window and the next message is undeleted and has not been examined (i.e., it is marked with a ‘‘?’’). If ALWAYS, then it will display the next undeleted message even if it has already been seen. Initally, T. T is roughly Laurel semantics, ALWAYS is Hardy semantics.

LAFITEMOVETOCONFIRMFLG

Controls whether Lafite requires confirmation of the Move To command. If ALWAYS, all moves require confirmation; if LEFT, then only left-button moves (selecting the destination from a menu) require confirmation; if MIDDLE, then only middle-button moves (using the ‘‘default Move To’’ folder) require confirmation; if NIL, then no moves require confirmation. Initially ALWAYS.

LAFITEBROWSERREGION, LAFITEDISPLAYREGION, LAFITEEDITORREGION

These are REGIONs which are used to describe where the primary (i.e. first) window of each type is to be placed on the screen. Initally, they are set to something ‘‘reasonable’’ for the standard initial display configuration. If you set them to `NIL` then you will be asked to specify a region (via `GETREGION`) the first time any such window is created.

`LAFITEDISPLAYFONT`, `LAFITEEDITORFONT`, `LAFITEHARDCOPYFONT`

These are the fonts used for displaying messages, composing messages, and making hardcopy. You may change them individually. They should be `FONTDESCRIPTOR`'s as returned by `FONTCREATE`. Initially, they are all TimesRoman12.

`LAFITEBROWSERFONT`

The font used for displaying the table of contents in the browser window. Initially, Gacha10.

`LAFITEMENUFONT`

The font used for the items in all Lafite menus. Initially Helvetica10 Bold.

`LAFITETITLEFONT`

The font used for the title bar (‘‘Lafite’’) in the Lafite status window. Initially Helvetica12 Bold.

`LAFITESTATUSWINDOWPOSITION`

Specifies where the status window appears when Lafite is invoked. It is a `POSITION` or `NIL` (in which case you will be asked to specify a position when Lafite starts).

`LAFITEOUTBOXSIZE`

Specifies the number of delivered messages that should be retained in your outbox. As you send more messages, older ones fall off the end. Increasing this number gives you a longer ‘‘history’’ from which you can select and re-edit old messages, but this desire should be balanced with the knowledge that you are tying down the resources used by each of those messages. Setting `LAFITEOUTBOXSIZE` to zero or `NIL` disables the outbox feature: after delivery, messages completely vanish. `LAFITEOUTBOXSIZE` is initially 2.

`LAFITENEWMAILTUNE, LAFITEGETMAILTUNE`

(Dandelion only) These are lists of the form acceptable to the function `PLAYTUNE`, or `NIL`, in which case they are ignored. `LAFITENEWMAILTUNE` is played when Lafite discovers you have new mail waiting; `LAFITEGETMAILTUNE` is played when a Get Mail command completes.

`LAFITEENDOFMESSAGESTR`, `LAFITEENDOFMESSAGEFONT`

`LAFITEENDOFMESSAGESTR` is a string containing the text of the ‘‘End of Message’’ token displayed at the end of a message; `LAFITEENDOFMESSAGEFONT` is the font in which it is displayed. If `LAFITEENDOFMESSAGESTR` is `NIL`, then no ‘‘End of Message’’ token will appear.

`LAFITEIFFROMMETHENSEENFLG`

If true, then messages sent from you are considered ‘‘Seen’’ (and hence do not have the mark ‘?’), even though you have not yet displayed them. Initially `T`.

`LAFITEBUFFERSIZE`

The number of 512-character buffers used by the stream managing the file behind an open browser window. If you regularly receive very long messages, you might want to increase this to improve performance of Display followed by HardCopy or Move. Initially 20, which handles up to about 10,000-character messages.

`LAFITEMODEDEFAULT`

Determines the mode Lafite comes up in when more than one set of protocol implementations has been loaded. Initially `NIL`, which means that if there is a choice, Lafite simply waits for the user to choose a mode.

`LAFITESHOWMODEFLG`

Determines whether Lafite displays the current mode in the status window. If `ALWAYS`, it always does; if `NIL`, it never does; if `T`, it does only if there is any ambiguity (more than one set of protocol implementations has been loaded). Initially, `T`.

Adding New Message Forms to Lafite

The normal way to add new message forms to Lafite is to edit an existing form (or build one from scratch) and save it away using the ‘Save Form’ menu item. You can also provide message forms that compute the text on the fly, as, for example, the ‘Lisp Support’ selection does. To add your own items to the ‘Message Forms’ menu, add a standard three-element menu item to the variable `LAFITESPECIALFORMS` and then set the variable `LAFITEFORMSMENU` to `NIL` (this is where the menu is cached). The three-element menu item should yield a `LITATOM` as its ‘‘value’’, that atom being interpreted as follows:

1. If the atom has a function definition, the function is called (with no arguments) and the returned value (a string or a TEdit TextStream) is used;

2. If the atom has a value, its value (a string or a TEdit TextStream) is used;

3. otherwise, a copy of the file by that name is used.

For example, if TEdit wanted to add a message form that contained the date the TEdit was made (similiar to Lafite’s bug report form) it could add

```
("TEdit Support" (QUOTE MAKETEDITFORM)
             "Make a form to report a problem with TEdit")
```

to `LAFITESPECIALFORMS`; `MAKETEDITFORM` could be defined to be

```
[LAMBDA NIL
  (PROG (OUTSTREAM)
        (SETQ OUTSTREAM (OPENTEXTSTREAM ""))
        (printout OUTSTREAM "Subject: TEdit: >>Subject<<" T)
        (printout OUTSTREAM "To: " TEDITSUPPORT T)
        (printout OUTSTREAM "cc: " (USERNAME) T)
        (printout OUTSTREAM "TEdit-System-Date: " TEDITSYSTEMDATE T T)
        (printout OUTSTREAM ">>Message<<" T)
        (RETURN OUTSTREAM]
```

where `TEDITSUPPORT` and `TEDITSYSTEMDATE` are variables set by TEdit. Lafite supplies one function to make this kind of message form easier to construct:

`(MAKEXXXSUPPORTFORM SYSTEMNAME ADDRESS SYSTEMDATE)`

Creates a message form (a TEdit stream) to be mailed to the maintainers of `SYSTEMNAME`. `SYSTEMNAME` is the name of the subsystem (a string); `SYSTEMDATE`, if non-`NIL`, is a date (string) of importance to include in the message; and `ADDRESS` is the mail system address of the intended recipient(s).

For example, if `MAKETEDITFORM` were defined as

```
[LAMBDA NIL
  (MAKEXXXSUPPORTFORM "TEdit" "TEditSupport" TEDITSYSTEMDATE)]
```

Then selecting ‘‘TEdit Support’’ in the Message Forms menu would produce a form such as the following:

```
Subject: TEdit: >>Subject<<
To: TEditSupport
cc: vanMelle.pa
TEdit-System-Date:  3-Feb-84 12:23:49
Lisp-System-Date:  3-Feb-84 18:13:22
Machine-Type: Dorado

>>Message<<
```