CARNEGIE-MELLON UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

SPICE PROJECT

Mercury

Mike Horowitz, Dave Nichols, Ed Smith

16 August 1984

Spice Document S167

Location of machine-readable file: [x]/usr/spicedoc/manual/spiceuser/hg

Copyright © 1984 Carnegie-Mellon University

This is an internal working document of the Computer Science Department, Carnegie-Mellon University, Schenley Park, Pittsburgh, Pennsylvania 15213 USA . Some of the ideas expressed in this document may be only partially developed, or may be erroneous. Distribution of this document outside the immediate working community is discouraged; publication of this document is forbidden.

Supported by the Defense Advanced Research Projects Agency, Department of Defense, ARPA Order 3597, monitored by the Air Force Avionics Laboratory under contract F33615-81-K-1539. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Projects Agency or the U.S. Government.

Table of Contents

1 Introduction

Mercury is the electronic mail system for Spice. Mercury provides commands for reading mail, composing new mail and organizing old mail. Mercury under Spice is almost identical to Mercury under Unix[1] and is very similar to the RdMail system on Tops-10 systems. The Spice system provides a transport mechanism known as Mailman for getting electronic mail to and from a Perq. The Mailman program is described in the *Spice Commands and Utilities Manual.* Mailman is run as a background server and, except for initialization, will not bother the user.

1.1 Conventions

Mercury is a typescript oriented program, so commands and their arguments must be typed on a single line terminated by a *<return>*. Commands and many keywords can be abbreviated as long as the abbreviation is unique to Mercury. Mercury will complain, of course, if it does not understand your intentions. We will use **bold face** to indicate information typed by Mercury and *italics* to indicate something typed by a user. Note that though Mercury comes from a Unix environment, it is *not* case-sensitive.

Mercury and its accomplice Mailman assume they are the only processes running that can access your mail. Therefore, it is *foolish and dangerous* to run multiple instances of Mercury or Mailman on a single Spice Machine or to run multiple instances of Mailman on different Spice Machines for the same user.

Mercury uses the file *mail.hg* to store mail. This file is kept in the identical format used by Mercury under Unix. In particular, you can transfer your *mail.hg* file from your Vax area to use on your Spice Machine. (Be sure to ship the file in *BINARY* mode!)

New mail arrives from Mailman and is put in the file *<boot>perqinbox*. Outgoing mail is written into files that can be matched in a directory scan as *<boot>perqoutbox**. If you see any of these files in your *<boot>* area it is best to leave them alone unless you know what you are doing.

2 How to Get Started

To start Mercury, type:

mercury

Mercury will identify itself, tell you the number of messages found in your *mail.hg* file and give the number of new messages delivered since you last checked. Mercury will then give you the so-called "command prompt":

1 TM, Bell Labs

Hg>

Mercury uses several different prompts. The command prompt indicates that you can type ordinary Mercury commands. Other prompts, detailed below, accept a different set of commands and will *not* accept ordinary Mercury commands.

3 How to Get Out

(subtitle: Sorcerer's Apprentice Section) To get out of Mercury, give the following command to Mercury at the command prompt:

Hg> *quit*

This command will get you back to the Spice Shell. This command, along with other ways of getting out of Mercury, are detailed below.

4 How to Get Help

Once inside Mercury, various help facilities are available. Typing:

Hg> *?*

will get you a list of commands and keywords used by Mercury. (This list was optimized to fit on precisely the screen of a *Concept 100*, and is thus fairly dense.) You can give the *help* command with an argument specifying what you want information about, and Mercury will oblige, often giving you more than you wanted. For instance, typing:

Hg> *help quit*

will tell you all about the *quit* command. Note that if there are several different topics that match the argument to the help command, you will be prompted for each topic. This prompt will ask for one of:

[Yes, No, Abort]

indicating whether you want to see this topic (answer *yes*), skip this topic and go to the next (answer *no*) or whether you have had enough and want to return to the **Command prompt** (answer *abort*).

Another help command is *syntax*. This command gives you the syntax that Mercury expects for a particular command. The following would give you the syntax for the *quit* command.

Hg> *syntax quit*

5 Basic Commands

Mercury maintains a list of all your messages and numbers them sequentially. Each message can be referenced with this *message number.* A *message sequence* is any combination denoting a series of message numbers or a message number. A *message sequence keyword* is an attribute of the messages, such as answered or deleted. The basic commands for reading and sending mail take either a message number sequence or a message keyword as its argument. A message sequence can be one of the following:

- empty, meaning the *current message* (your current message is the number of the last message in your mail file upon first entering mercury)
- a single message number, meaning just that message (e.g. 57)
- a sequence of message numbers separated by commas, indicating those messages (e.g. 63, 25, 75)
- a message number, followed by an exclamation mark, followed by an integer *count,* indicating that message and *count* number of messages following (e.g. 37 !4 which is equivalent to 37, 38, 39, 40)
- a message number, followed by a colon, followed by another message number, indicating the messages from the first number to the second, inclusive (e.g. 37:41)
- the keyword *all,* indicating all messages (e.g. all)
- any message sequence keyword (explained later in the header command paragraph).

A *header* is one line of information containing various characteristics about a message or group of messages. A header has the format of the following example:

41?- + 30 Oct 83 John.Doe@Spice...foo(267)

This header indicates the following (from the left):

- **29** - the message number
- **?-+** - various flags indicating message attributes (described later with the *headers* command)
- **30 Oct 83 John.Doe@Spice** - the date of creation of the message and the sender's name
- **foo** - the Subject of the message
- **(187)** - the length, in characters, of the message.

The commands to be explained in the reading mail section are checkmail, current, type, previous, next, delete, expunge, undelete, exit, kill, headers. The commands in the sending mail section are mail, retry, answer, forward, and remail. The full details will be described in the more advanced commands section.

5.1 Reading Mail

When you start Mercury, it announces whether you have any new mail. For each piece of new mail, Mercury gives you a *header*. For example, Mercury might type the following when it first comes up:

```
[One new message found in mail box: 29]
29?- +30 Oct 83 John.Doe@Spice... foo (187)
Hg>
```

At any time in Mercury, the user can check on the existence of new mail with the *checkmail* command:

```
Hg> checkmail
[One new message found: 34]
Hg> checkmail
[No new mail found in mail box.]
```

If you type the *current* command, Mercury will show the message number of the current message, the total number of messages, and the mail file that is open.

```
hg> current
current message is 23 of 23 messages: mail.hg
```

To read an entire message, the user types the *type* command:

```
Hg> type 29
—- Message 29 is —-
Date: 30 Oct 83, 11:24:22
From: John.Doe@Spice
To: You
Subject: foo

Hi. Bye.
Hg>
```

Two other ways of typing messages are the *next* command and the *previous* command which type the next or previous message, respectively. An example (assuming the current message number is 3):

```
Hg> previous
—- Message 2 is —-
Date: 30 Oct 83, 11:24:22
From: John.Doe@Spice
To: You
Subject: foo

How's the Mercury document coming?
```

```
Hg> next
—- Message 4 is —-
Date: 30 Oct 83, 11:24:22
From: John.Doe@Spice
To: You
Subject: foo

Just checking.
Hg>
```

To delete a message, the user types the *delete* command:

```
Hg> delete 29
```

The *delete* command only marks messages to be deleted when the user executes the *expunge* command. Doing an *expunge* command will show you the headers of all messages to be deleted and prompt you for what to do.

```
Hg> expunge
 29?- +30 Oct 83 John.Doe@Spice foo (187)
Expunge deleted messages? [Yes, No, Abort]:
```

If you say *yes*, all messages marked for deletion will be deleted from the mail file and destroyed. If you say *no* or *abort*, the messages will not be deleted. Note that when you destroy messages with *expunge*, the messages that are left are renumbered.

The user can "undelete" a deleted message with the *undelete* command:

```
Hg> undelete 29
```

In the section above on how to get out of Mercury, we mentioned the *quit* command. Another way of getting out is to use the *exit* command.

```
Hg> exit
 4?- +30 Oct 83 John.Doe@Spice foo (187)
Expunge deleted messages? [Yes, No, Abort]:
```

Notice that the *exit* command performs an *expunge* command before leaving Mercury. Answering *yes* will destroy all deleted messages and leave Mercury. Answering *no* will simply leave Mercury. Answering *abort* will not destroy any messages and not leave Mercury. You will be returned to the command prompt.

A useful shorthand is the *kill* command, which does a *delete* followed by a *next*. So, if the current message is 3, then executing the *kill* command will delete message 3 and type message 4, like so:

```
Hg> kill
—- Message 4 is —-
Date: 30 Oct 83, 11:24:22
From: John.Doe@Spice
```

```
To: You
Subject: foo

Just checking.
Hg>
```

This is particularly useful when going through the morning's mail, deleting the message you just read and showing you the next one. Getting into the habit of typing this to read all your newest mail can unfortunately delete mail you wish to keep: be careful!

The *headers* command can be used to type the headers for the message or messages indicated.

```
Hg> headers 3
3 * John.Doe@Spice... foo (145)
```

In particular, using the keyword *all* with the headers command:

```
Hg> headers all
1   John.Doe@Spice... more foo (153)
2   John.Doe@Spice... foo (512)
3 * John.Doe@Spice... foo (145)
...
```

prints the headers for all messages in the user's mail file. Each header has a set of flags printed after the message number. These flags have the following meaning:

- ? the message has not been answered (see the *answer* command below on sending mail)
- + the message is new, (that is, this message was read in from the mailbox during this run of Mercury)
- - the message has not been examined (that is, typed)
- * the message has been marked for deletion
- B this is a *blind* copy of a message (described below on sending mail)
- D this is a draft of message (described below on sending mail).

Each of these flags can be used as message sequence keyword. For instance, to see the headers for all deleted messages, type:

```
Hg> headers deleted
 3 * John.Doe@Spice... foo (145)
```

Or, to type all answered mail, type:

```
Hg> type answered
```

5.2 Sending Mail

There are various ways to create and send mail with Mercury. The basic command is just:

Hg> *mail*

The *mail* command takes as an optional argument the name of the recipient of the message. This command runs the editor, which under Spice is Oil. Upon invoking the editor, the screen will display a *mail template* which initially looks something like the following:

```
Date: 30 Oct 83, 11:31:22
From:
To:
cc:
Subject:
Message-ID: <1983.10.30.11.31.22.mumble>

<< Put body of message here >>
```

When you fill in the template, it is essential to fill in the **From** and **To** fields in order for Mercury to deliver your mail. You can leave the **cc** field (a remarkable holdover from the days when typewriters struck paper and could make "carbon copies") blank. Typing the original mail command as "mail Joe" creates a template whose **To** field is already filled in with "Joe". In order to let Mercury know where the message heading ends and your message begins, be sure to *leave a blank line between the Message-ID and the first line of your message!* Using the Spice editor, remove the line about the "body of message" and include your message there.

After exitting the editor normally, you will then be back running Mercury. Mercury will then give you the "action prompt", which looks like this:

Action (ABort, Blind, Draft, Edit, Mail, Type):

The prompt expects one of the following:

- *Abort* - this will return you to the command prompt, destroying the message you created
- *Blind* - this will create a blind copy of the message, place it in your mail file, and return to the action prompt
- *Draft* - this will create a draft copy of the message, place it in your mail file, and return to the command prompt
- *Edit* - this will return you to the editor to work some more on your message
- *Mail* - this will actually mail your message, returning you to the command prompt
- *Type* - this will type your message, and then return you to the action prompt

Note that several of the above options return you to the action prompt. Thus, you can make a blind copy of the message, type it out, and then mail it before returning to the command prompt. Draft messages are mail that Mercury assumes you will eventually get back to editing again.

The *retry* command on a draft message puts you in the action prompt for that message, allowing you to edit it some more or to mail it. So if message 6 is a draft of a message, then you can do a retry command on it, and you will see the following:

```
Hg> retry 6
Action (ABort, Blind, Draft, Edit, Mail, Type):
```

A message can be retried (i.e. re-edited) as many times as desired as long as you type *Draft* at the action prompt. As with all the other basic commands, *Retry* can be given a message sequence of more than one number. Mercury will succesively prompt for each message. Draft messages are deleted after they are mailed. Note you cannot use the *mail* command with a message number to send draft mail since the *mail* command will interpret the message number as the recipient of a new message template!

Another way to create mail is to use the *answer* command. This command, followed by a message sequence allows you to "answer" messages. That is, it creates a message template whose **To** field is a list containing the names of the senders of all the messages in the sequence, the **cc** field is a list of all people who received copies of the original mail, and the **Subject** field is a line begining with "Re:" followed by the subject of the first message in the sequence. Except for these changes in the message template, this commands acts like the *mail* command. Of course, as with all commands that require a *message sequence*, you can leave it blank to act on the current message. The following command will answer the current message:

```
Hg> answer
```

You can forward a message with the *forward* command. This command takes as its arguments a message sequence, optionally followed by a slash character (/) and a list of recipients. This acts much like the *mail* command. The difference is that the message template created for you to edit will contain a copy of the message you are forwarding. So, executing the forward command:

```
Hg> forward
```

might create a template like the following:

```
Date: 30 Oct 83, 11:31:22
From:
To:
cc:
Subject:
Message-ID: <1983.10.30.11.31.22.mumble>

<< Put body of message here >>
— Begin forwarded message —
Date: 30 Oct 83, 11:31:22
From: John.Doe@Spice
```

```
To: You
cc:
Subject: foo
Message-ID: <1983.10.30.11.31.22.mumble>

Hi.
--- End forwarded message ---
```

A faster way of forwarding mail, especially when you don't need to add any text to the message, is the *remail* command. This command takes as its arguments a message sequence, a slash character (/), and a list of recipients. If the list of recipients is omitted, the command prompts for them.

```
Hg> remail 3/John.Doe@Spice
```

6 Advanced Commands

This section details more advanced features of Mercury. Included in this section are more details about messages and message headers as well as the various commands to organize and search through messages.

6.1 Conventions

Mercury uses a couple of environment variables to find some information about the user. The environment variable *MailAddress* is looked up when Mercury initializes to find the mail address of the user. The value of this variable is used to fill in the **From** field in message templates. You should make this something like <VaxAccount>CMU-CS-SPICE or wherever you have mail sent. If this value is null, then the **From** field will always be blank. The user must fill in this field or Mercury will refuse to deliver the mail. This field should contain something that a recipient of your mail could use to send a reply to.

Another environment variable looked up by Mercury is *MailFile.* The value of this variable is taken as the name for the user's default mail file. If this value is null, then the name "mail.hg" is used. Note that no specific directory is given, so the search path will be used to find this file. The user can change mail files at any time with one of the commands *read* or *qread,* described below in the section on mail files.

6.2 Message Headers

Each message contains a large amount of information in its header. The header is that part of the message at the top that contains various fields such as **Date**, **From**, **To**, and so on. You can see the entire header of a message with the *wholeheaders* command:

```
Hg> wholeheader 3
 3 + 7 Nov 83 John.Doe@Spice  foo (154)
Date: 7 Nov 83  1039 EST (Monday)
From: John.Doe@Spice
To: You
Subject: foo
Message-Id: <07Nov83.103922.mumble>
```

Another command to see information about a message is the *whereis* command. This takes a message sequence and returns all sorts of atributes and parameters kept by Mercury about your messages:

```
Hg> whereis 3
 3 + 7 Nov 83 John.Doe@Spice  foo (154)
  System attributes(s): Answered Examined New
  Class name(s): foo
```

Class names are discussed in a subsequent section on using mail files as a data base.

6.3 Mail Files

Mercury provides various mechanisms for organizing mail. One way is with multiple mail files. When Mercury first comes up it looks for your default mail file. This file is found in the environment variable *MailFile*. (If *MailFile* is null, then the name "mail.hg" is used.) You can create a new, empty mail file with the *create* command. This command takes the name of the mail file you want to create and creates it:

```
Hg> create newmail.hg
[Creating newmail.hg]
```

Of course, if you try to create a mail file that already exists, Mercury will complain.

You can change mail files within Mercury with the *read* command. This command takes the name of a mail file as its only argument. The current mail file is exitted:

```
Hg> read old>mail.hg
[There are 20 messages in >sys>spice>old>mail.hg]
```

Note that the *read* command does an *exit* on the current mail file before reading in the new one. If you wish to *quit* out of the current mail file instead of *exit* (that is, you do *not* want to do an *expunge* of deleted messages), then do a *qread* command.

To put messages into a mail file from the current mail file, use the *put* command. This command takes a message sequence and a mail file name (separated by a slash, "/") and creates copies of the messages in the mail file. Mercury tells you the current number of messages in the mail file before it starts, then the number of messages when it's done:

```
Hg> put 3/newmail.hg
[There are zero messages in newmail.hg]
[There are now one messages in newmail.hg]
```

Another similar command is the *move* command. This takes the same arguments as *put*, but marks the messages moved for deletion in the current mail file. These messages will of course be deleted during the next expunge.

6.4 Classification of Messages

Another way of organizing messages is to use message classes within an individual mail file. First, to create a message class, you use the *allocate* command:

```
Hg> allocate myclass
```

To see a listing of all classes, use the *display* command:

```
Hg> display
myclass 0   Unclassified  4
```

Note that there is an implicit class of "unclassified" messages. The number to the right of the class name is the number of messages in each class.

To add a message to a class, you use the *classify* command. This command takes a message sequence, followed by a slash, followed by a class name, like so:

```
Hg> classify 1,2,3/myclass
```

Doing a *display* command after this reveals what you would expect:

```
Hg> display
myclass 3   Unclassified  1
```

Note that the number of messages in all classes will *not* necessarily add up to the total number of messages in the mail file. This is because a message may belong to more than one class:

```
Hg> alloc anotherclass
Hg> classify 1/anotherclass
Hg> display
anotherclass   1 myclass   3 Unclassified 1
Hg> headers all
 1  7 Nov 83  John.Doe  foo1 (153)
 2  7 Nov 83  John.Doe  foo2 (134)
```

```
3  7 Nov 83  John.Doe  foo3 (255)
4  7 Nov 83  John.Doe  foo4 (231)
```

A quick way of seeing all messages that have not yet been classified (to me, that means messages I haven't dealt with yet), do a *headers not classified* command.

Messages can be removed from classes with the declassified command:

```
Hg> declassify 1/anotherclass
```

Classes are destroyed with the *deallocate* command:

```
Hg> deallocate anotherclass
```

If you attempt to deallocate a class that is not empty, Mercury will complain. A quick way to declassify all messages in a class is:

```
Hg> declassify anyclass / anyclass
```

This creates a message sequence of all messages in the class *anyclass* and deletes all of those messages from the class *anyclass*.

6.5 Using Mail Files as Data Bases

Mail files are actually maintained as data bases of messages and related information. You can, for instance, substitute for any message sequence an expression that evaluates to a message sequence. This expression can test message attributes, class names, fields of messages headers (such as **From**, **Subject**, **To**, **CC**, etc.), and so on. This expression can includes unions, intersections, subtractions and negations. The expression can use the following attribute names to match messages: **Answered**, **Blind**, **Deleted**, **Draft**, **Examined**, **Kept**, **New**, **Newest** or **Private**. The expression can also use the following message sequence keywords to match messages: **All**, **BackReference**, **Before**, **BodySearch Classified**, **Context**, **During**, **FieldSearch**, **Handled**, **HeaderSearch**, **Last**, **MessageSearch**, **NextLast**, **Processed**, **Reference**, or **Since**.

The best reference to such expressions is in the help facilities provided within Mercury. Help is provided about "message sequence", "system attributes" and "set expressions" that should explain more than you will want to know about using this facility.

An interesting command to use with this mechanism is *context*. This will establish a sub-set of messages in the current mail file that are visible. Mercury will change its prompt (by adding an extra greater-than sign) to indicate that you are nested within a context. This command can be executed again, within a context, to restrict even further the messages visible to the user. An example:

```
Hg> context since 7-nov-83
[3 messages]
Hg>>
```

```
context myclass
[1 message]
Hg>>>
```

To get out of a context, execute a *pop* command. This will leave the current context and return to the next higher context. Mercury's prompt will change indicating the change in context.

7 Summary of Commands

The following sections gives a brief summary of the commands in the order described above.

7.1 Getting Started

- *quit* - exits Mercury immediately; no messages marked for deletion are actually deleted;
- *?* - gives a brief summary of all commands and keywords known to Mercury;
- *help <command-or-keyword>* - takes a command name or keyword as its argument and types out much help information about the topics in Mercury that match the argument;
- *syntax <command>* - takes a command name for its argument and returns a brief description of the syntax and arguments expected for the command;

7.2 Basic Commands

- *current* - shows you some information about the "current" message, including its number and the mail file currently open (default is *mail.hg*);
- *checkmail* - checks on new messages in the mail box; if there are any, Mercury reads them and assigns them message numbers;
- *type <message-sequence>* - types the messages specified; if no message is specified, this types the current message;
- *next* - types the message after the current message;
- *previous* - types the message preceding the current message;
- *delete <message-sequence>* - marks for deletion the specified messages; if no message is specified, then this deletes the current message;
- *expunge* - destroys all messages that have been marked for deletion;
- *undelete <message-sequence>* - undeletes the specified messages (that is, removes the mark for deletion for each message in the *<message sequence>*; this command will *not* restore a message destroyed by the *expunge* command;
- *kill* - performs a *delete* command on the current message, followed by a *next* command;
- *headers <message-sequence>* - types a header for the messages specified;
- *mail <recipient>* - lets the user edit a message template, then upon return from the editor, delivers the message to the recipient;
- *retry* <message-sequence> - allows the user to retry a draft message;

- *answer <message-sequence>* - answers the specified messages; this command works just like the *mail* command, obtaining the recipient of this message from the source of the message(s) being answered;
- *forward <message-sequence>/<recipient>* - allows the user to forward mail to another recipient; the mail being forwarded is included as a part of the message template;
- *remail <message-sequence>/<recipient>* - is a short version of the *forward* command; this command simplies mails the messages to the recipients specified without letting the user edit the messages before they are delivered;

7.3 Advanced Commands

- *wholeheaders <message-sequence>* - types the entire message header for each of the messages given;
- *whereis <message-sequence>* - types system attributes for all messages given;
- *read <mail-file>* - exits from the current mail file (doing an expunge of all deleted messages) and reads in the given mail file;
- *qread <mail-file>* - reads in the given mail file, doing a *quit* of the current mail file;
- *create <mail-file>* - creates a new, empty mail file of the given name;
- *put <message-sequence> / <mail-file>* - puts the specified messages in the specified mail file;
- *move <message-sequence> / <mail-file>* - puts the specified messages in the specified mail file, marking the messages for deletion in the current mail file;
- *allocate <class-name>* - allocates a new class of the given name;
- *display <class-name>* - displays all classes and the number of messages in each class;
- *classify <message-sequence> / <class-name>* - adds messages to a specified class;
- *declassify <message-sequence> / <class-name>* - removes messages from a specified class;
- *deallocate <class-name>* - destroys a class;
- *context <message-sequence>* - establishes a context out of the specified messages;
- *pop* - returns from the current context to the next higher context;

7.4 Action-Prompt Commands

The following are possible answers to the action prompt:

- *Abort* - cancel the command that is awaiting action;
- *Blind* - create a blind copy of the message involved and return to the action prompt;
- *Draft* - create a draft copy of the message involved and return to the command prompt;
- *Edit* - enter the editor on the message involved; Mercury will display the action prompt again when the user returns from the editor;
- *Mail* - mail the message and return to the command prompt;
- *Type* - type the message and return to the action prompt;