To: Distribution

From: Steve Herbst

Subject: New message commands

Date: June 1, 1974

INTRODUCTION

MTB-070 described two secure mail commands called mail and send_mail (*). This MTB proposes some commands for secure interactive messages. Secure in the latter case means not only that the messages are protected but that the process of someone who is accepting messages cannot be interrupted by an unauthorized user. This means that the event channel the accepter sets up is only available to a sender who has proper extended access to his ring 1 mailbox.

Sending a message should be similar to sending a piece of mail, but there are a few differences. One is that interactive communication between consoles operates best on a line-by-line basis. The line is a natural unit in Multics (eg. the command line) and the sentence is a natural unit for people. The end of a line is a good place to interrupt.

Some of the features of mail and send_mail are inappropriate for one-line messages. These are:

1) sending a segment

2) using edm_

3) deleting by number

Other features, such as reading selectively, sending by name only and forwarding, are directly applicable to one-line messages. The remaining distinction is that interactive messages are accompanied by wakeups.

These wakeups can be handled various ways. Most important is the amount of control each user has over when his process can be interrupted and how much work another process can make it do. Wakeups are divided into two classes, normal and urgent. The user can defer normal messages while accepting urgent ones. Which class of wakeup accompanies a message is at the discretion of the sender.

(*) indicates terms defined in MTB-070.

Here are some ways the recipient can exercise control over wakeups:

1) The "w" and "u" (normal and urgent) extended-access bits in the acl of his mailbox grant different wakeup permission to different users. The two bits are independent. If a particular user is overdoing the urgent messages, an acl entry can give him "w" but no "u" access.

2) "allow_normal" and "allow_urgent" bits in the mailbox header can be turned off to defer wakeups from all users. If "allow_normal" is on, "allow_urgent" must be on.

3) A "-rdy" option when accepting messages causes the printing of a message to be deferred until just before the next ready message.

4) No wakeup can be sent to a user who has not explicitly accepted messages.

5) No wakeup can be sent to a user who is not logged in.

If someone is unable to send a wakeup he can still put his one-line message in the mailbox. Interactive messages share <mailbox_name>.mbx with mail. The "wakeup" bit in mail_format (*) tells "messages" which messages to print, and the "urgent" bit tells "messages -urgent" which messages to print.

THE COMMANDS

While ipc_message_facility continues to operate using <Person>.con_msgs, a program called "messages" will use ring 1 mailboxes. Since the two facilities are independent, a user will have to accept both kinds of messages or a sender will have to try one when the other fails. Furthermore, programs that use the process id (such as the uninstalled "apt" command which prints information in the Active Process Table entry) may have to look elsewhere than con_msgs.

The following commands are entries in "messages":

```
messages -path- -options-
```

prints interactive messages in path or path.mbx. If path is not specified it assumes the user's default mailbox and if the default mailbox does not exist, it creates one with an acl giving "wu" to *.*.*. (accept and defer do the same.) Options are:

-urgent	print and delete only urgent messages
-all	print old and new messages

Normally, messages prints only messages whose "has_been_read" bit (*) is off, turning the bit on afterwards. If the "has_been_read" bit is already on and the date-time is more than 24 hours old, the message is deleted. "-all" indicates that all the messages are to be printed and none is to be deleted.

accept -path- -options-

creates an event-call channel if one does not already exist, naming wakeup_messages_ as the entry to call. It stores the channel id and process id in the header of the specified mailbox. (The assumption is that there is at most one process to wake up per mailbox.) It sets the "allow_normal" and "allow_urgent" bits in the mailbox header. Finally, it calls messages_ to print anything received at an earlier time. Options are:

-urgent	accept and print only urgent messages
-rdy	defer printing until the next ready message
-hold	do not turn on the "has_been_read" bit

messages, accept and wakeup_messages_ call a subroutine messages_ that prints interactive messages. messages_ calls read_mail_ (*) repeatedly, performs deletion and sends acknowledgements.

defer -path- -options-

defers certain wakeups by turning off the "allow_normal" and/or "allow_urgent" bits. Channel and process id's remain in the mailbox header. Options are:

-urgent	defer all wakeups instead of just normal wakeups
-rdy	same as before

The "-rdy" option with accept and defer is implemented as follows:

turn on "defer_till_ready", an internal static flag.

call cu_$get_ready_proc (ready_entry) to save the current ready procedure in internal static.

call cu_$set_ready_proc (ready_messages_), where ready_messages_ is an entry in "messages".

The next time accept or defer is used without the "-rdy" option, these effects are reversed.

wakeup_messages_ returns without doing anything if "defer_till_ready" is on. ready_messages_, which is now the ready procedure, calls messages_, calls ready_entry, and returns.

send destination -options- text1 ... textn

sends one or more lines. destination has the form Person.Project, or Person to be looked up in mail_table (*). Either Person or Project can be "*" but not both. Operators might be exceptionally allowed to specify *.* or * as a destination so they can use this command to broadcast deferrable messages (as opposed to "warn" messages, which are not deferrable).

If any texti's exist they are concatenated with single spaces between them to form the message. Otherwise, send types "Input:" and sends each input line until it encounters a line consisting solely of a period. Becauwe a wakeup is sent for each line, an entire conversation can be carried on with a single invocation of the command.

We want to duplicate send_mail's interface but are forced to depart from it slightly. The interface suggested for send includes the following simplest case:

send x message

This is the easiest way to send one line to one person. If send allowed multiple destination arguments as send_mail does, it would have to know where the text of the message begins. Moving the message out of the command line or surrounding it with quotes would make the command more cumbersome to use. A control argument "-text" to be inserted when there is more than one destination, ie.

send arg1 ... argn -text text1 ... textn

where argi is a destination or an option, could cause a bad message to be sent if it were left out. In this MTB, multiple destinations are specified using the "-ds" control argument.

send takes the following options:

-urgent	send an urgent message
-silent or -sl	do not inform the sender if the message cannot be sent

-brief or -bf	give minimal information
-acknowledge or -ak	ask the recipient's message_ to send back an acknowledgement when it prints the message
-list	(*) the next argument is the absolute or relative path name of a mailing list
-destination or -ds	the next argument is one additional destination

When send cannot send to one of its destinations, it tells the user why on the next line, before "Input:" if applicable. Possible reasons are:

1) "Entry not found. <path>.mbx"

   a message cannot be sent if there is no mailbox to keep it in.

2) "<explanation>"

   explanation comes from <Person>.fwd (*) and says that Person will not be reading his messages. Depending on how <Person>.fwd is implemented, the nature of a query will vary.

3) "<Person> is not logged in."

   QUERY: "Anyway?"

4) "<Person> is not accepting messages."

   QUERY: "Anyway?"

5) "<Person> is deferring all messages."

   QUERY: "Anyway?"

6) "<P¬erson> is deferring messages."

   QUERY: "Is it urgent?"

   QUERY (if answer was "no"): "Anyway?"

7) "<error> Cannot add the message."

8) "<error> Cannot send a wakeup."

QUERY: "Anyway?"

9) "<error> Message was added but no wakeup."

<error> is a standard error message for a code returned by a ring 1 primitive. The query "Anyway?" can be answered one of three ways:

"yes"	put a message in the mailbox without sending a wakeup. (If no text was given, type "Input:")
"no"	quit. (Any text in the command line disappears.)
"urgent"	put an urgent message in the mailbox without sending a wakeup.

"-silent" prevents the above warnings and queries, in which case the default for 1), 2), 7) and 9) is to do nothing and for the others is to add the message to the mailbox. If an unusual error has occurred, a message will be printed regardless of "-silent".

"-brief" provides the same defaults but prints either "OK", "no wakeup" or "nothing sent".

A NEW PRIMITIVE

Wakeup protection necessitates a protected channel id. The channel and process id's in the ring 1 mailbox can never be passed to ring 4. Therefore, a primitive in ring 1 has to send all the wakeups. In MTB-070 it was suggested that a command to send interactive messages use the ring 4 primitive send_mail_ to add each message to the mailbox. It is probably more efficient for a single ring 1 call to add the message, send the wakeup, and deal with the various combinations that arise.

The send command locates the recipient's mailbox (diagnostic 1 above) and calls gate entry mailbox_$wakeup which calls mbx_mseg_$wakeup:

```
call mbx_mseg_$wakeup (index, ptr, bitcnt,
                       urgent, result, code);
```

where index is the index of the mailbox, urgent is either on or off, and result is a fixed bin output describing the first unsuccessful attempt to do something. When mbx_mseg_$wakeup assigns a value to result, it returns immediately:

if the mailbox's forwarding bit is on, result = 2

```
if the recipient is not logged in, result = 3

if channel id or process id is missing, result = 4

if both "allow_normal" and "allow_urgent" are off, result = 5

if urgent and "allow_normal" are off and "allow_urgent" is  on,
result = 6

if no access to send wakeup, result = 8

add the message; if an error occurs, result = 7

send the wakeup; if an error occurs, result = 9

return
```

Depending on the value of result and the user's answer to a query, send may wish to call mailbox_$add_index.

ACKNOWLEDGEMENT

An acknowledgement tells the sender that his message has been read. His message might get read immediately after a wakeup, or at some later time. By this time the active call channel of the sender may be different or nonexistent, and so may his process. For this reason, all the checks that "send" makes have to be made again to send the acknowledgement.

message_ looks at the "acknowledge" bit in the message's mail_format_. If the bit is on, messages_ locates the sender's mailbox and calls mailbox_$wakeup to send a standard acknowledgement:

"Message received <Person>.<Project> <date & time>

It should send only one acknowledgement to each sender. If a wakeup cannot be sent, the acknowledgement still goes in the mailbox and if this is also impossible, message_ forgets the whole thing.

APPENDIX

Data in the mailbox header:

```
channel_id fixed bin(71)
process_id bit(36) aligned
allow_normal bit(1)
allow_urgent bit(1)
```

Data in the message (mail_format):

```
wakeup bit(1)
urgent bit(1)
has_been_read bit(1)
acknowledge bit(1)
```

Data in "messages" internal static:

```
ready_entry entry
defer_till_ready bit(1)
hold bit(1)
```