Multics Task Report MTR-064

To: Distribution

From: Steve Herbst

Subject: Implementation of secure mail and messages

Date: 10/1/74

With the current mail facility, a user has to have "rw" access to a mailbox in order to add a message. He is then able to do whatever he pleases with and to the contents of the mailbox. Secure mail uses a ring 1 message segment for a mailbox and interprets extended access for each user.

The same problem exists for interprocess messages added to a ring 4 con_msgs segment. Wakeups are an additional problem. When a user sets up a process id and channel id to receive wakeups, anyone with "r" access to the con_msgs segment can send a wakeup to his process. Extended access has separate fields to control wakeups. For convenience, secure interprocess messages will occupy the same mailbox as secure mail.

The Access Isolation Mechanism enhances the ring 1 solution to secure mail and messages. In the enhanced system, every individual message has an access class. Different class messages can occupy the same ring 1 (within the security kernel) mailbox. The access class of a message and the extended access on the mailbox determine access to the message.

New commands will be installed to send and receive secure mail and interprocess messages. While users become accustomed to these new interfaces, both new and old commands will work. In the case of mail, the current command will be converted to use ring 1 mailboxes and the new interface will simply provide an alternate way to send secure mail. In the case of interprocess messages, the current commands will continue to work with user-ring con_msgs segments and the new interface will use ring 1 mailboxes.

Ultimately, the following commands will exist:

MAIL	MESSAGES	
send_mail	send	
mail	messages	(to read)
	accept	
	defer	

A subroutine called send_mail_ will send messages with or without wakeups. All of these use the same ring 1 mailbox in the user's home directory with the name <Person>.mbx.

The following commands will be phased out:

mail (sending capability)

send_message	print_messages
send_message_silent	short_message_format
send_message_acknowledge	long_message_format
accept_messages	unlock_messages
defer_messages	imf_state

The following segments will be phased out:

<Person>.con_msgs
mailbox

The conversion process has five stages:

Phase 0 (already begun)

Exposure of a new mail command with the same interface as the old one. Users of >udd>m>lib>e>mail have new mailboxes created automatically to coexist with the old ones. These users will have to delete their old mailboxes after Phase 1. Meanwhile, they must use the standard mail command or >udd>m>lib>e>old_mail to read what other users put in the old mailboxes. Commands are installed in the standard system to manipulate extended access on new mailboxes and to create, delete and rename new mailboxes (see Appendix). >udd>m>lib>e>mail sends the old way when there is no new mailbox.

Phase 1 (Dec 1, System 2.1)

Installation of a new mail command in the standard system. "mail" the first time not only creates a new mailbox but deletes the old one. This phase gets most people converted to new mailboxes. To prevent being converted, the user must either keep something in "mailbox":

```
Old mailbox contents:
        etc.
Delete? no
```

or turn on its safety switch. In either case, a new mailbox is not created and mail continues to go to the old one in ring 4. This is one way to keep using private mail reading programs that need real access to the mailbox. Alternately, the user can rewrite his mail reading programs to call mailbox_ gate entries for reading messages from a new mailbox. These entries are documented in the forthcoming PLM on Message Segments, Order Number AN69.

Phase 2 (Jan 1, System 2.2?)

Installation of new interfaces for sending mail:

1) send_mail_ is a subroutine that sends one piece of mail to one user defined by a person_id and a project_id. If project_id is blank, send_mail_ calls mail_table_manager_$lookup, which returns a default mailbox pathname for person_id. This pathname resides in an administrator-maintained table called mail_table. Installation of the segment mail_table and the program mail_table_manager_ is part of Phase 2. Procedures are provided to create mail_table from existing system tables and to update the information in it automatically when users are added to or removed from the system. MTB-122 describes mail_table and the interfaces for mail_table_manager_ and send_mail_.

2) send_mail is a command whose interface is different from that of mail. The new interface is designed to accomodate certain new features. The send_mail command handles these features and calls send_mail_.

"send_mail" is like "mail *", ie. it sends console input, unless the control argument "-segment" is present. A destination for send_mail and for the send command of Phase 3 is of the form Person.Project, where Project is usually missing. The average sender does not care where a person's mailboxes are located. The default mailbox listed in mail_table handles this kind of communication. send_mail_ interprets the blank project_id as a request to call mail_table_manager_$lookup.

Phase 3 (Feb 1, System 2.2?)

Installation of the commands send, accept and defer that implement interprocess messages in the new mailboxes. See MTB-085 for descriptions of these new interfaces. The send command calls send_mail_ to add messages and send wakeups.

Phase 4 (July 1, System 3.1?)

Removal of the old interfaces. Only one set of commands should be supported. ipc_message_facility_ will be phased out as users delete their con_msgs segments.

Appendix

The following info segments exist now:

mail_changes.info	new_mail.info
mbx_commands.info	
mbx_create.info	mbx_delete.info
mbx_add_name.info	mbx_delete_name.info
mbx_rename.info	mbx_delete_acl.info
mbx_list_acl.info	mbx_set_acl.info

The following MTB's exist on related subjects:

070	5/1/74	New mail commands
085	6/1/74	New message commands
122	10/1/74	Mail subroutines