TO: Distribution

FROM: Joan Archer Scott

DATE: 26 November 75

RE: Multics Change Requests

Enclosed are copies of Multics Change Requests which were approved from 1 November 75 through 15 November 75.

Ver. 3
741022 MULTICS CHANGE REQUEST MCR 1464

TITLE: Fix bug in operator DOWN command

AUTHOR: Paul Green

STATUS | DATE
Written | 10/20/75
Status | A 11/04/75
Expires | 04/20/76

Planned for System: not applicable
Fixes Bug Number(s): MPRF 8753
Documented in MTB: not applicable
Incompatible Change: no
User/Operations-visible Interface Change: no
Coded in: (X)PL/I ( )ALM ( )other-see below
Performance: ( )better (X)same ( )worse

DOCUMENTATION CHANGES (specify one or more)
MPM (vol,sect) MPAM (sect)
MOSN (sect) MSAM (sect)
PLMs (AN#)
Info Segs
Other
None (reason) no change to documentation

CATEGORY (check one)
( )Lib. Maint. Tools
( )Sys. Anal. Tools
( )Sys. Prog. Tools
( )355
( )BOS
( )Salvager
( )Ring Zero
( )Ring One
(X)SysDaemon/Admin
( )Runtime
( )User Command/Subr

OBJECTIONS/COMMENTS:

Headings are: SUMMARY, REASONS, IMPLICATIONS, DETAILED PROPOSAL (optional)

SUMMARY:

Fix the operator DOWN command to always prevent further logins.

REASONS:

The operator DOWN command behaves improperly if all users are already logged out when the scheduled shutdown occurs. It returns before shutting off further logins. Thus, users can still login after the system thinks everyone is out.

IMPLICATIONS:

Unattended operation will be more reliable.

DETAILED PROPOSAL:

Replace admin.pl1 in bound_user_control_

Ver. 3
741022 MULTICS CHANGE REQUEST

MCR 1465

TITLE: Rewrite ring 4 syserr log management

AUTHOR: L. Scheffler

STATUS	DATE
Written	10/24/75
Status	R 11/04/75
Expires	04/24/76

Planned for System: MR 3.1
Fixes Bug Number(s): not applicable
Documented in MTB: 103
Incompatible Change: yes
User/Operations-visible Interface Change: yes
Coded in: (X)PL/I ( )ALM ( )other-see below
Performance: (X)better ( )same ( )worse

CATEGORY (check one)
( )Lib. Maint. Tools
(X)Sys. Anal. Tools
( )Sys. Prog. Tools
( )355
( )BOS
( )Salvager
( )Ring Zero
( )Ring One
( )SysDaemon/Admin
( )Runtime
( )User Command/Subr

DOCUMENTATION CHANGES (specify one or more)
MPM (vol,sect) MPAM (sect)
MOSN (sect) MSAM (sect)
PLMs (AN#) AN65
Info Segs
Other

OBJECTIONS/COMMENTS:

Headings are: SUMMARY, REASONS, IMPLICATIONS, DETAILED PROPOSAL (optional)

SUMMARY

1. Redo ring 4 syserr logging using vfile_, and delete log_util_.

2. Create module syserr_log_util_ in bound_admin_tools_ to provide a virtual interface for programs searching the syserr log. Rewrite print_syserr_log and daily_syserr_process to use this module.

REASONS

1. vfile_ was ruled out for the initial implementation of ring 4 syserr logging because of two vfile_ disabilities. First, deleted record space was not recovered. Date-deletion of old syserr messages would require rewriting the entire log. Second, the seek operation required an exact key match. This made searching for messages in approximate time ranges difficult. These disabilities have been removed, and vfile_/msf_manager_ are superior to log_util_.

2. The specific structure of the syserr log should be isolated in a single module. This frees current and planned future syserr log analysis/summary tools from details that are better handled centrally.

IMPLICATIONS

1. Code size is reduced (by about 700 lines). A more common, better tested, and more appropriate mechanism is used.

2. Code in print_syserr_log, daily_syserr_process, and future tools is reduced and simplified.

3. Ring 4 syserr logs currently maintained (only at MIT, CISL, and Phoenix), will be incompatible. (The ring 4 syserr logging mechanism is disabled in MR3.0.) A program will be provided to convert any currently maintained ring 4 syserr log segments to the new (vfile_) format prior to installation.

DETAILED PROPOSAL

1. See MTB-103 and MCRs 820, 1144 and 1438 for details of the current implementation of ring 4 syserr logging.

   The basic change to the ring 4 syserr log mechanism is to replace calls to the module log_util_ with the equivalent iox_ calls.

   The permanent ring 4 syserr log will be a single indexed-sequential file named >sc1>perm_syserr_log. Each syserr message will be a single record in this file. The key of each record will be a character string representing the date and time the message was logged (to microsecond precision), the message sequence number, and its syserr action and sorting codes. The ASCII ordering of keys will also be the date-time order or messages. No interpretation of the internal structure of syserr messages will be done when inserting or reading them: they will be referenced simply as bit strings. This leaves the exact format of syserr messages as a convention only between syserr_logger (hardcore) and the syserr log tools.

   The internal structure of a syserr message will be described in a new include file syserr_message.incl.pl1, to be used by ring 4 tools reading the syserr log. syserr_log.incl.pl1 (defining the syserr log and message structure in ring 0) will be changed to include syserr_message.incl.pl1 to define message structure.

2. A new module syserr_log_util_ will contain entry points to open, search (by message time), position (N messages forward or back), read (a single message), and close the "syserr log", consisting of the ring 4 (vfile_) syserr log, and the ring 0 log. All tools reading the syserr log should use these entries. The open entry will open the ring 4 log, copy (via audit_gate_) the ring 0 log into the process directory, and construct a new temporary vfile_ in the process directory containing only those messages from the ring 0 log not yet in the ring 4 log. Search and position operations then view the two vfiles as one continuous log. All this is necessary because 1) users with access to read the syserr log should not (generally) be able to write it; and 2) during normal Multics operation the ring 0 syserr log generally contains messages not yet in the ring 4 log. If the process does not have access to either the ring 0 or ring 4 logs, only the portion of the log that is accessible will be used. print_syserr_log and daily_syserr_process will indicate if the caller does not have access to part of the log.

   See the attached description of syserr_log_util_ for details.

Name: syserr_log_util_

This module provides entry points to read the syserr log. All programs that read the syserr log should use this module, rather than directly accessing either the ring 4 or ring 0 syserr logs.

Entry: syserr_log_util_$open

This entry "opens" the syserr log for reading. This entry must be called before any other entries in this module are called.

Usage

```
dcl syserr_log_util_$open entry (bit (36) aligned, fixed bin
    (35));

call syserr_log_util_$open (access, code);
```

where

1. access — indicates which of the permanent (ring 4) and current (ring 0) logs are accessible to the caller. (Output)

   "00"b — no access to any part of the syserr log
   "10"b — access to permanent log only
   "01"b — access to current log only
   "11"b — access to both and current logs

2. code — is a status code (Output)

Notes

If both the permanent (ring 4) and current (ring 0) logs are accessible, code is 0. If either log is not accessible, code is error_table_$moderr, and access indicates which of the current and/or permanent logs are accessible. If the syserr log is currently being updated, code is error_table_$file_busy. If the syserr log is already open, code will be error_table_$not_closed.

Entry: syserr_log_util_$close

This entry "closes" the syserr log after reading.

Usage

```
dcl syserr_log_util_$close entry (fixed bin (35));

call syserr_log_util_$close (code);
```

where code is a status code (Output) and is either 0, or error_table_$not_open, if the syserr log was not previously open.

The following entries deal with a single syserr "message", whose structure is defined in syserr_message.incl.pl1. The search and position entries set the "current message". The read entry reads the current message set by the previous search or position. After the syserr log is opened, but before any calls to search or position, the current message is undefined.

Entry: syserr_log_util_$search

This entry searches for the oldest syserr message whose time of creation is greater than or equal to the specified time, and sets the current message to the message found.

Usage

```
dcl syserr_log_util_$search entry (fixed bin (71), fixed bin
     (71), fixed bin (35), fixed bin (3), fixed bin (5),
     fixed bin (17), fixed bin (35));

call syserr_log_util_$search (search_time, time, seq,
     action, sort, length, code);
```

where

1. search_time — is the message time for searching, in microseconds. (Input)
2. time — is the actual time the message found was logged, in microseconds. (Output)
3. seq — is the syserr log sequence number of the message found. (Output)
4. action — is the syserr action code of the message found. (See Syserr Action Codes in this volume.) (Output)
5. sort — is the syserr sorting class s (See Syserr Sorting Codes in this volume.) (Output)

6. length — is the total length of the message found, in 36-bit words. (Output)

7. code — is a status code. (Output)

Notes

If search_time is the special value 0, the current message is set to the oldest (earliest) syserr message, and code is 0. If search_time is the special value -1, the current message is set to the newest (most recent) syserr message, and code is 0. If search_time is earlier than the earliest syserr message, the current message is set to the earliest message, and code is 0. If search_time is later than the latest syserr message, the current message is set to the latest message, and code is error_table_$end_of_info. If search_time is invalid (less than -1), code is error_table_$bad_arg.

Entry: syserr_log_util_$position

This entry moves a given number of messages forward or backward from the current syserr message (the last message located by either the search or position entries), and returns that message. If this is the first call to either the position or search entries after opening, positioning is from the end of the syserr log. (i.e., a position of -1 as the first position operation after opening sets the current message to the last syserr message in the log.)

Usage

```
dcl syserr_log_util_$position entry (fixed bin (24), fixed
    bin (71), fixed bin (35), fixed bin (3), fixed bin (5),
    fixed bin (17), fixed bin (35));

call syserr_log_util_$position (n, time, seq, action, sort,
    length, code);
```

where

1. n — is the number of messages forward ($n \geq 0$) or backward ($n < 0$) to move. (Input)

2. time — is as above. (Output)

3. seq — is as above. (Output)

4. action is as above. (Output)

5. sort is as above. (Output)

6. length is as above. (Output)

7. code is a status code. (Output)

Note

If there are less than abs (n) messages between the current message and the end (n>0) or start (n<0) of the syserr log, code is error_table_$end_of_info and the current message is set to the latest (n>0) or earliest (n<0) syserr message.

Entry: syserr_log_util_$read

This entry reads the body of the current syserr message (located by a previous call to search or position). The current message is not changed.

Usage

```
dcl syserr_log_util_$read (pointer, fixed bin, fixed bin,
     fixed bin (35));

call syserr_log_util_$read (bufp, bufl, messl, code);
```

where

1. bufp is a pointer to a buffer into which the current syserr message will be put. (Input)

2. bufl is the length of the buffer, in 36-bit words. (Input)

3. messl is the actual length of the message, in 36-bit words. (Output)

4. code is a status code. (Output)

Note

If the current message has not been set by a previous call to search or position, messl is 0, and code is error_table_$no_record. If the supplied buffer is smaller than the syserr message being returned, messl is 0 and code is error_table_$long_record.

Multics Change Request

MCR 1466
Page 1 of 1

TITLE: Fix to tape_ for LA firmware

AUTHOR: B. Silver

STATUS	DATE
Written	10/21/75
Status	A 11/04/75
Expires	05/04/76

DOCUMENTATION CHANGES

Document	Specify One or More
MPM (Vol, Sect.)	
PLMS (AN #)	
MOSN (Sect.)	
MPAM (Sect.)	
MSAM (Sect.)	
Info Segs	
Other (Name)	
None (Reason)	none needed

-Coded in: [X] PL/I [ ] ALM [ ] other- explain in DETAILED PROPOSAL
-Planned for System MR ____
-Fixes Bug Number(s) ____
-Documented in MTB ____
-User/Operations-visible Interface change? [ ] yes [X] no
-Incompatible change? [ ] yes [X] no
-Performance: [ ] Better [X] Same [ ] Worse
-Replaces MCR ____

	Category (Check One)
	Lib. Maint. Tools
	Sys. Anal. Tools
X	Sys. Prog. Tools
	355
	BOS
	Salvager
	Ring Zero
	Ring One
	SysDaemon/Admin.
	Runtime
	User Cmmd/Subr.

Objections/Comments:

Use these headings: SUMMARY, REASONS, IMPLICATIONS, DETAILED PROPOSAL (Optional)

SUMMARY:

Change tape_ to better handle MPC Data Alert status.

REASONS:

Tape MPC firmware changed to return MPC Data Alert (major) Preamble Error (sub) status instead of Device Data Alert (major) Lateral Parity Error (sub).

DETAILED PROPOSAL:

Change tape_read_ to treat MPC Data Alert errors as non-fatal. Change tdcm_ to count MPC Data Alert errors along with Device Data Alert errors.

Ver. 3
741022

Multics Change Request

MCR 1471
Page 1 of 1

TITLE: Fix bug in abbrev .l request

AUTHOR: S. Herbst

STATUS	DATE
Written	10/27/75
Status	A 11/04/75
Expires	05/04/76

-Coded in: [X] PL/I [ ] ALM [ ] other- explain in DETAILED PROPOSAL
-Planned for System MR
-Fixes Bug Number(s)
-Documented in MTB
-User/Operations-visible Interface change? [ ] yes [X] no
-Incompatible change? [ ] yes [X] no
-Performance: [ ] Better [X] Same [ ] Worse
-Replaces MCR

	Category (Check One)
	Lib. Maint. Tools
	Sys. Anal. Tools
	Sys. Prog. Tools
	355
	BOS
	Salvager
	Ring Zero
	Ring One
	SysDaemon/Admin.
	Runtime
X	User Cmmd/Subr.

DOCUMENTATION CHANGES

Document	Specify One or More
MPM (Vol, Sect.)	Commands Volume
PLMS (AN #)	
MOSN (Sect.)	
MPAM (Sect.)	
MSAM (Sect.)	
Info Segs	
Other (Name)	
None (Reason)	doc ok

Objections/Comments:

Use these headings: SUMMARY, REASONS, IMPLICATIONS, DETAILED PROPOSAL (Optional)

SUMMARY:

1. Fix bug in abbrev that prevents the .l request from printing abbreviation values longer than 256 characters.

2. Update abbrev to iox_.

3. Fix bug that sometimes causes storage to be freed twice when using .r mode.

Note: The MPM documentation mentions an obsolete restriction that the value of an abbreviation cannot be longer than 132 characters. This restriction should be deleted from the writeup (see attached).

Control Requests

Before abbrev expands a command line (to pass it on to the normal command processor), it first checks to see if the command line is an abbrev request line. An abbrev request line has a period (.) as the first nonblank character of the line. Any command line interpreted as an abbrev request line is treated specially and is neither checked for embedded abbreviations nor (even in part) passed on to the normal command processor. The one exception to this rule is a command line with a space character following the period; the rest of the line is passed to the normal command processor without being expanded.

The character immediately after the period of an abbrev request line is the name of the request. The following requests are recognized:

.a <abbr> <rest of line>	add the abbreviation <abbr> to the current profile. It is an abbreviation for <rest of line>. Note that the <rest of line> string can contain any characters. If the abbreviation already exists, the user is asked if he wishes to redefine it. The user must respond with "yes" or "no". The abbreviation must be no longer than eight characters and must not contain break characters. The string it stands for must be no longer than 132 characters.
.ab <abbr> <rest of line>	add an abbreviation that is expanded only if found at the beginning of a line or directly following a semicolon (;) in the expanded line. In other words, this is an abbreviation for a command name.
.af <abbr> <rest of line>	add an abbreviation to the profile and force it to overwrite any previous abbreviation with the same name. The user is not asked if he wants the abbreviation redefined.
.abf <abbr> <rest of line>	add an abbreviation that is expanded only at the beginning of a line and force it to replace any previous abbreviation with the same name. The user is not asked if he wants the abbreviation redefined.
.d <abbr1> ... <abbrn>	delete the specified abbreviations from the current profile.
.f	enter a mode (the default mode) that forgets each command line after executing it. See the .r and .s requests.
.l <abbr1> ... <abbrn>	list the specified abbreviations with the strings they stand for. If no abbreviations are specified, all abbreviations in the current profile are listed.

er. 3
41022

Multics Change Request

MCR 1458
Page 1 of 1

TITLE: Restrict max length to page boundaries

AUTHOR: S. Herbst

STATUS	DATE
Written	10/21/75
Status	CPA 11/11/75
Expires	05/11/76

-Coded in: [X] PL/I [ ] ALM [ ] other- explain in DETAILED PROPOSAL
-Planned for System MR
-Fixes Bug Number(s)
-Documented in MTB
-User/Operations-visible Interface change? [X] yes [ ] no
-Incompatible change? [X] yes [ ] no
-Performance: [ ] Better [X] Same [ ] Worse
-Replaces MCR

	Category (Check One)
	Lib. Maint. Tools
	Sys. Anal. Tools
	Sys. Prog. Tools
	355
	BOS
	Salvager
	Ring Zero
	Ring One
	SysDaemon/Admin.
	Runtime
X	User Cmmd/Subr.

DOCUMENTATION CHANGES

Document	Specify One or More
MPM (Vol, Sect.)	SWG hcs_$set_max_length
PLMS (AN #)	
MOSN (Sect.)	
MPAM (Sect.)	
MSAM (Sect.)	
Info Segs	pending_changes.info
Other (Name)	
None (Reason)	

Objections/Comments:

Use these headings: SUMMARY, REASONS, IMPLICATIONS, DETAILED PROPOSAL (Optional)

SUMMARY:

Change hcs_$set_max_length to restrict the setting of max length to whole pages. Currently, it rounds the desired max length up to zero mod 16 without returning an error code. The same behavior is proposed, rounding up to zero mod 1024 instead. Modify the warning message printed by the set_max_length command when it rounds up to say 1024 instead of 16.

REASONS:

Cures problem where hcs_$truncate_seg gets a segfault trying to zero the last page of a segment if the max length is in the middle of that page.

IMPLICATION:

Incompatible change that should not hurt anybody.

Ver. 4
750508

Multics Change Request

MCR 1470
Page 1 of 1

TITLE: Fix bug in do active function

AUTHOR: S. Herbst JWG

STATUS	DATE
Written	10/27/75
Status	P A 11/11/75
Expires	05/11/76

DOCUMENTATION CHANGES

-Coded in: [X] PL/I [ ] AIM [ ] other- explain in DETAILED PROPOSAL
-Planned for System MR
-Fixes Bug Number(s)
-Documented in MTB
-User/Operations-visible Interface change? [X] yes [ ] no
-Incompatible change? [X] yes [ ] no
-Performance: [ ] Better [X] Same [ ] Worse
-Replaces MCR

	Category (Check One)
	Lib. Maint. Tools
	Sys. Anal. Tools
	Sys. Prog. Tools
	355
	BOS
	Salvager
	Ring Zero
	Ring One
	SysDaemon/Admin.
	Runtime
X	User Cmnd/Subr.

Document	Specify One or More
MPM (Vol, Sect.)	
PLMS (AN #)	
MOSN (Sect.)	
MPAM (Sect.)	
MSAM (Sect.)	
Info Segs	
Other (Name)	
None (Reason)	doc ok

Objections/Comments:

Use these headings: **Summary of Proposal, Reasons for Proposal, Implications, Detailed Proposal.**

SUMMARY:

Change the do command to not check for control arguments when called as an active function.

REASONS:

Control arguments change do's mode of operation and are intended to be used in a command line of the form:

```
do control_args
```

The active function usage of do, however, returns a command line as its value and should return control args in the command line without alteration.

Currently, if the command line argument to the do active function begins with a minus sign, it is incorrectly taken as an invalid control argument to do.

Ver. 3
741022

Multics Change Request

MCR 1472
Page 1 of 2

TITLE: Implement compatible secure message commands

AUTHOR: S. Herbst

STATUS	DATE
Written	10/23/75
Status	A 11/11/75
Expires	05/11/76

-Coded in: [X] PL/I [ ] AIM [ ] other- explain in DETAILED PROPOSAL
-Planned for System MR ____
-Fixes Bug Number(s) ____
-Documented in MTB ____
-User/Operations-visible Interface change? [ ] yes [X] no
-Incompatible change? [ ] yes [X] no
-Performance: [ ] Better [X] Same [ ] Worse
-Replaces MCR ____

	Category (Check One)
	Lib. Maint. Tools
	Sys. Anal. Tools
	Sys. Prog. Tools
	355
	BOS
	Salvager
	Ring Zero
	Ring One
	SysDaemon/Admin.
	Runtime
X	User Cmnd/Subr.

DOCUMENTATION CHANGES

Document	Specify One or More
MPM (Vol, Sect.)	AG92
PLMS (AN #)	
MOSN (Sect.)	
MPAM (Sect.)	
MSAM (Sect.)	
Info Segs	
Other (Name)	
None (Reason)	

Objections/Comments:

Use these headings: SUMMARY, REASONS, IMPLICATIONS, DETAILED PROPOSAL (Optional)

SUMMARY:

Implement compatible secure versions of send_message, print_messages, and other commands in the ipc message facility, to use the ring 1 Person.mbx mailboxes instead of ring 4 Person.con_msgs segments.

REASON:

To make use of extended access in a secure message facility, as was done for the mail command.

DETAILED PROPOSAL:

Compatibility implies sending the old way when the recipient does not have a new mailbox (though most users do by now), printing the contents of Person.con_msgs when invoked to print new style messages, and accepting messages stored in either place.

The commands involved are:

send_message
print_messages
accept_messages
immediate_messages
defer_messages
send_message_silent
send_message_acknowledge
long_message_format
short_message_format

The mail command also has to be changed to print send_message messages saved in the mailbox. This change will make the count ("n messages") that mail prints be correct, and is natural because send_message messages are usually higher priority than mail. Messages from send_message are printed in brief format plus line numbers, and groups of lines from the same sender are preceded by:

message from <sender> (date-time)

preemption_time	if the user is a primary user, returns the time at which he becomes eligible for group preemption. The time is of the form "hhmm.t".
brief_bit	returns true if the user specified the -brief control argument in his login line; otherwise, returns false.
protected	if the user is currently a primary user and protected from preemption, returns true; otherwise, returns false.
absin	if the user is an absentee user, this returns the absolute pathname of his absentee input segment including the absin suffix; otherwise returns a null string.
absout	if the user is an absentee user, returns the absolute pathname of his absentee output segment; otherwise, returns a null string.

Example

The following example illustrates the use of one of the active functions described abcve.

```
ioa_ [user login_time]
```

This example causes the time the user logged in to be printed at the user's terminal.

Name: have_mail

or pending interactive messag

The have_mail active function returns the value true if there is mail in the user's current default mailbox or in a specified mailbox; otherwise, false is returned.

Usage

```
have_mail -path-
```

where path is the pathname of a mailbox. If path is not specified, the have_mail active function looks at the user's ring 1 default mailbox. If this mailbox does not exist, the active function looks at the user's ring 4 default mailbox. If neither mailbox exists, the active function returns the value false.

accept_messages

Name: accept_messages, am

The accept_messages command initializes or reinitializes the user's process for accepting messages sent by the send_message command. If the ~~segment~~ mailbox:

>udd>Project_id>Person_id>Person_id.~~con_msgs~~ mbx

does not exist, the accept_messages command creates it. A channel is created to receive wakeups from send_message so that when a message is received, it is printed on the user's terminal immediately. ~~Once the con_msgs segment is created,~~ Messages sent when the user is not logged in or when he is deferring messages (see the defer_messages command) are saved in the ~~con_msgs segment. (To see the messages that have been saved in the segment, invoke the print_messages command.)~~ mailbox & can be read later by invoking the print_messages command. The mail command stores mail in the same mailbox. For a description of mailbox access, see "Extended Access" below.

Usage

accept_messages -control_args-

where the control_args can be chosen from the following list:

-brief, -bf prevents accept_messages from informing the user that it is creating a ~~con_msgs segment~~ mailbox. This control argument also causes messages to print in short format (see the -short control argument below).

-long causes every message printed to be preceded by the sender's Person_id and Project_id. This is the default mode.

-print prints all messages that were received since the last time the user was accepting messages.

-short causes repeated messages from the same sender to be preceded by "=:" instead of the Person_id and Project_id.

Notes

The user should not create the con_msgs segment in any other way except by invoking the accept_messages command.

The user should not give conflicting control arguments in the same invocation of the command (i.e., -long and -short or -long and -brief).

Channel and process identifiers are stored in the ~~con_msgs segment~~ user's mailbox. Since only one process can receive a wakeup when a message is placed in the segment, it is not advisable for several users to share the same ~~con_msgs segment~~ mailbox.

Extended Access

Access on a newly created mailbox is automatically set to adrosw for the user who created it, as for *.SysDaemon.*, and aow for *.*.*.

The types of extended access are:

add	a - add a message
delete	d - delete any message
read	r - read any message
own	o - read or delete only your own message, that is, those sent by you.
status	s - find out how many messages are in the mailbox
wakeup	w - send a normal priority wakeup

Mailboxes created by the mail command give adros access to the creator and ao to *.*.*. The first time the accept_messages command is invoked, w access is added to every entry that already has a access except the entry for *.SysDaemon.*. Access on a mailbox can be changed using the commands mbx_set_acl and mbx_delete_acl, and listed using the command mbx_list_acl.

Name: send_message, sm

The send_message command sends messages (one or more, always sent one line at a time) to a given user on a given project. The messages are placed in the mailbox:

>udd>Project_id>Person_id.[redacted]mbx

If such a segment does not exist, the messages are placed in:

>udd>Project_id>anonymous_messages>Person_id.con_msgs

For a description of the mailbox, refer to the accept_messages command.

If the recipient is accepting messages (see the accept_messages and defer_messages commands), send_message sends wakeups, causing his process to print each message immediately on his terminal.

Usage

send_message Person_id Project_id -message-

where:

1. Person_id is the registered name of the recipient.
2. Project_id is the name of the recipient's project.
3. message is an optional string that can be up to 132 characters long. If message is missing from the command line, send_message types "Input." and accepts a variable number of lines that it sends, one line at a time, with each newline character. In this case, input is terminated by a line consisting solely of a period.

Notes

Parentheses, quotes, brackets, and semicolons in the command line have their usual command language interpretation. This means, for example, that:

sm Person_id Project_id testing complete; installation this week

sends:

testing complete

and prints an appropriate error message (probably "Segment installation not found.") because the characters typed after the semicolon are another command line.

Name: defer_messages, dm

The defer_messages command prevents messages sent by the send_message command from printing on the user's terminal. Instead, these messages are saved in the user's ~~can_mail segment~~ mailbox. For a description of the mailbox, refer to the accept_messages command.

Usage

defer_messages

Notes

The print_messages command prints messages that have been deferred.

If the user wants to restore the printing of messages on his terminal, he can invoke the immediate_messages command.

Name: print_messages, pm

The print_messages command prints any interprocess messages that were received (and saved in the user's mailbox) while the user was not accepting messages. ~~(For a description of the con_mega segment refer to the accept_messages command.)~~ For a description of the mailbox, refer to the accept_messages command.

Usage

print_messages

Notes

The user must issue either the accept_messages or defer_messages command before issuing the print_messages command.

If messages are deferred, it is a good practice to print out pending messages periodically.

Name: mail, ml

The mail command allows the user to send a message to another user or to print messages in any mailbox to which he has sufficient access. The extended access used on mailboxes permits the creator of a mailbox to firmly control other users' access to his mailbox. Adding, reading, and deleting messages are independent privileges under extended access. For example, one user can be given access to only add messages, and another user to add messages and also read and delete only the messages he has added. For more information on extended access, see "Creating a Mailbox" below. Mail sent to a user is placed in the mailbox named >user_dir_dir>Project_id>Person_id>Person_id.mbx in his home directory.

Usage

To send mail:

mail path Person_id1 Project_id1 ... -Person_idn- -Project_idn-

where:

1. path — is the pathname of a segment to be sent or is an asterisk (*) to indicate that the user wishes to type a message to be sent (see "Composing Mail" below).
2. Person_idi — is the name of a person to whom mail is to be sent.
3. Project_idi — is the name of a project on which Person_idi is registered.

To print ~~mail~~ messages sent by the mail+send_message commands:

mail -path- -control_arg-

where:

1. path — is the pathname of a mailbox. If the mbx suffix is not given, it is assumed. If no path argument is given, the contents of the default mailbox is printed (see "Creating a Mailbox" below).
2. control_arg — can be -brief or -bf so that only the total number of messages in the mailbox is printed. If the mailbox is empty, nothing is printed.

Ver. 3
741022 MULTICS CHANGE REQUEST MCR 1473

TITLE: Make default search rules installation changeable.

AUTHOR: VanVleck

STATUS | DATE
Written | 10/22/75
Status | A 11/17/75
Expires | 04/22/76

Planned for System: not applicable
Fixes Bug Number(s): not applicable
Documented in MTB: not applicable
Incompatible Change: no
User/Operations-visible Interface Change: no
Coded in: (X)PL/I ( )ALM ( )other-see below
Performance: ( )better (X)same ( )worse

DOCUMENTATION CHANGES (specify one or more)

MPM (vol,sect) MPAM (sect)
MOSN (sect) MSAM (sect) X
PLMs (AN#) 66
Info Segs
Other

CATEGORY (check one)
( )Lib. Maint. Tools
( )Sys. Anal. Tools
( )Sys. Prog. Tools
( )355
( )BOS
( )Salvager
(X)Ring Zero
( )Ring One
( )SysDaemon/Admin
( )Runtime
( )User Command/Subr

OBJECTIONS/COMMENTS:

Headings are: SUMMARY, REASONS, IMPLICATIONS, DETAILED PROPOSAL (optional)

SUMMARY: Add a new hphcs_ entry which will permit the system administrators to change the default search rules for all processes. Make a new tool which will call this gate entry and load the default search rules from a segment.

REASONS: The current default search rules are assembled into active_hardcore_data and cannot be changed. Making these rules a site default provides flexibility for sites with special library structures; in particular, it will allow us to experiment with optional libraries such as the "CISL>sss" on the development machine.

IMPLICATIONS: none

Name: set_default_search_rules

This highly-privileged command is used in the Initializer process to set the installation's default search rules for all processes.

Usage

```
set_default_search_rules path
```

1) path is the path name of a default search rules segment.

Default_Search_Rules_Segment

Each line in the default search rules segment may be either a keyword or the absolute pathname of a directory to be searched. The order of the lines in the default search rules segment gives the order in which the rules will be applied by a user process.

The legal keywords are:

```
initiated_segments
referencing_dir
working_dir
home_dir
process_dir
```

The absolute pathname rules may be tagged with one or more identifiers, which name a group of rules. A user process may specify the tag instead of specifying the entire list of directories containing that tag.

The maximum number of search rules which can be specified is a system constant. It is currently equal to 22.

Example

If the installation places the following lines in its default search rules segment, it will recreate the default rules used if set_default_search_rules was not called:

```
initiated_segments
referencing_dir
working_dir
>system_library_standard,system_libraries
>system_library_unbundled,system_libraries
>system_library_1,system_libraries
>system_library_tools,system_libraries
>system_library_auth_maint,system_libraries
```

Ver. 3
741022

MULTICS CHANGE REQUEST

MCR 1474

TITLE: Correct and complete various hardware description include files.

AUTHOR: EJ Wallman

STATUS DATE

Written 10/28/75

Status A 01/11/75

Expires 04/28/76

Planned for System: not applicable
Fixes Bug Number(s): not applicable
Documented in MTB: not applicable
Incompatible Change: no
User/Operations-visible Interface Change: no
Coded in: (X)PL/I (X)ALM ( )other-see below
Performance: ( )better (X)same ( )worse

DOCUMENTATION CHANGES (specify one or more)

MPM (vol,sect) MPAM (sect)
MOSN (sect) MSAM (sect)
PLMs (AN#) AL39 & AN87
Info Segs
Other

CATEGORY (check one)

( )Lib. Maint. Tools
( )Sys. Anal. Tools
( )Sys. Prog. Tools
( )355
( )BOS
( )Salvager
(X)Ring Zero
( )Ring One
( )SysDaemon/Admin
( )Runtime
( )User Command/Subr

OBJECTIONS/COMMENTS:

Headings are: SUMMARY, REASONS, IMPLICATIONS, DETAILED PROPOSAL (optional)

SUMMARY: This MCR proposes the upgrade of various processor hardware description include files to match the current hardware. The include files affected were found to be discprepant during prepartion of the text for the System Debugger's Handbook, PLM AN87, and the Multics Processor Manual, AL39. These two documents already reflect the proposed changes.

REASON: With the advent of Multics native mode T&D and closer attention to the details of error reporting and recovery with HEALS II, a correct and complete description of the hardware becomes mandatory as part of the standard system..

IMPLICATIONS: The modules listed under DETAILED PROPOSAL will have to be reviewed for naming conflicts and recompiled with the new include files.

DETAILED PROPOSAL: It is proposed that the following system include files be modified as shown (by output of compare_ascii)...

```
>ldd>include>mode_reg.incl.alm

B4          bool        mr.trap_opcode_match,000002    trap on opcode match flag
Inserted before:
A4          bool        mr.trap_address_match,000001   trap on address match flag

A6          bool        mr.trap_opcode_match,000200    trap on opcode match flag
Changed to:
B7          bool        mr.trap_cu_oflo,000200         trap on CUHR overflow flag
B8          bool        mr.strobe_cu_on_opcode,000100  strobe CUHR on opcode match

B10         bool        mr.test_mode,000010            TEST MODE switch setting
Inserted before:
A8          bool        mr.enable_mr,000001            enable the mode register (DL)

          Modules affected:   fim.mexp                 (hardcore)
                              ii.alm                   (hardcore)
                              wired_fim.alm            (hardcore)
                              page_fault.alm           (hardcore)

>ldd>include>mode_reg.incl.pl1

A15             2 pad1 bit(3),
Changed to:
B15             2 test bit(1),              /* test mode */
B16             2 pad1 bit(2),

A23             2 pad1 bit(1),
Changed to:
B24             2 stm bit(2),               /* set timing marigns */

A25             2 pad2 bit(1),
A26             2 stm bit(1),               /* set timing marigns */
A27             2 pad3 bit(9))unaligned;
A28
A29         /*      END INCLUDE FILE ... mode_reg.incl.pl1 */
Changed to:
B26             2 pad3 bit(10))unaligned;
B27
B28         /*      END INCLUDE FILE ... mode_reg.incl.pl1 */

          Modules affected:   NONE

>ldd>include>history_regs.incl.pl1

A162            2 apu_pad1 bit(7),
Changed to:
B162            2 apu_pad0 bit(3),
B163            2 cache bit(1),             /* segment is encacheable */
B164            2 apu_pad1 bit(3),

A185            2 apu_pad1 bit(7),
Changed to:
```

```
B187            2 apu_pad0 bit(3),
B188            2 cache bit(1),             /* segment is encacheable */
B189            2 apu_pad1 bit(3),

B218            2 bit bit(1),               /* single bit type inst. */
Inserted before:
A214            2 du_pad1 bit(4),

B282            2 bit bit(1),               /* single bit type inst. */
Inserted before:
A277            2 du_pad1 bit(4),
```

Modules affected: NONE

Ver. 3
741022

Multics Change Request

MCR 1475
Page 1 of 1

TITLE: Implement reprint_query command

AUTHOR: S. Herbst

STATUS	DATE
Written	11/03/75
Status	A 11/11/75
Expires	05/11/76

-Coded in: [X] PL/I [ ] ALM [ ] other- explain in DETAILED PROPOSAL
-Planned for System MR ____
-Fixes Bug Number(s) ____
-Documented in MTB ____
-User/Operations-visible Interface change? [X] yes [ ] no
-Incompatible change? [X] yes [ ] no
-Performance: [ ] Better [X] Same [ ] Worse
-Replaces MCR ____

	Category (Check One)
	Lib. Maint. Tools
	Sys. Anal. Tools
	Sys. Prog. Tools
	355
	BOS
	Salvager
	Ring Zero
	Ring One
	SysDaemon/Admin.
	Runtime
X	User Cmnd/Subr.

DOCUMENTATION CHANGES

Document	Specify One or More
MPM (Vol, Sect.) Commands	
PLMS (AN #)	
MOSN (Sect.)	
MPAM (Sect.)	
MSAM (Sect.)	
Info Segs	
Other (Name)	
None (Reason)	

Objections/Comments:

Use these headings: SUMMARY, REASONS, IMPLICATIONS, DETAILED PROPOSAL (Optional)

SUMMARY:

1. Implement the command reprint_query (rq) as part of the command_query_ module. This command repeats the last question asked by command_query_ if that question has not yet been answered, otherwise types "no pending question".

2. Remove program_interrupt handler from command_query_.

REASON:

command_query_'s pi handler does not uphold standard interpretation of program_interrupt condition. If the user wants to abort a program, even just after it has asked a question, he should be able to do so by invoking pi.

Name: reprint_query

This command reprints the last question asked by the console (by the command_query_ subroutine) if the user has not yet answered that question.

This command is useful for re-interpreting questions that are garbled.

Usage: reprint_query

Note: If no question has been asked or the latest question was answered, the error message "No pending query" occurs.

Example:

Suppose that the console starts to type a question while the user is typing input. The query looks like:

```
E@foo.pl1?
```

The user signals QUIT and invokes reprint_query. The console types:

```
Do you want to delete the old segment foo.pl1?
```

The user then types "start" and answers the query.

Ver. 3
741022

Multics Change Request

MCR 1476
Page 1 of 1

TITLE: Add features to do and exec_com

AUTHOR: S. Herbst

STATUS	DATE
Written	11/03/75
Status	A 11/11/75
Expires	05/11/76

-Coded in: [X] PL/I [ ] ALM [ ] other- explain in DETAILED PROPOSAL
-Planned for System MR ____
-Fixes Bug Number(s) ____
-Documented in MTB ____
-User/Operations-visible Interface change? [X] yes [ ] no
-Incompatible change? [ ] yes [X] no
-Performance: [ ] Better [X] Same [ ] Worse
-Replaces MCR ____

	Category (Check One)
	Lib. Maint. Tools
	Sys. Anal. Tools
	Sys. Prog. Tools
	355
	BOS
	Salvager
	Ring Zero
	Ring One
	SysDaemon/Admin.
	Runtime
X	User Cmmd/Subr.

DOCUMENTATION CHANGES

Document	Specify One or More
MPM (Vol, Sect.) Commands	
PLMS (AN #)	
MOSN (Sect.)	
MPAM (Sect.)	
MSAM (Sect.)	
Info Segs	
Other (Name)	
None (Reason)	

Objections/Comments:

Use these headings: SUMMARY, REASONS, IMPLICATIONS, DETAILED PROPOSAL (Optional)

SUMMARY:

Add the following argument substitution features to do and exec_com:

1. The string &fi (f means "from") stands for arguments from i through the last argument supplied.

2. The string &n stands for the number of arguments supplied.

REASONS:

The &f feature is useful where the number of arguments is not known in advance. Currently, the user has to predict the maximum number and explicitly specify that many arguments. The &n feature is useful for testing whether exec_com was called with the correct number of arguments. Also, &f&n is an easy way to reference the last argument.

do

Name: do

The do command expands a command line according to the arguments supplied following the command string. The expanded command line is then passed to the Multics command processor for execution. If abbreviations are being expanded in the user's process prior to invocation of the do command, any abbreviations in the expanded command line will also be expanded. (See the description of the abbrev command.) Control arguments can be used to print the expanded command line, suppress its execution, or pass it back as the value of an active function.

Each instance of the parameter designator &i is replaced by the actual argument control_arg i through the last argument supplied. Each instance of the string &n is replaced by the number of actual arguments supplied.

Usage

```
do "command_string" -control_args-
```

where:

1. command_string is a command line (in quotation marks). Each instance of the parameter designator &i (where i is a number from 1 to 9) found in command_string is replaced by the corresponding actual argument control_argi. If any control_argi is not supplied, then each instance of &i in command_string is replaced by the null string. Each instance of the unique-name designator &! found in command_string is replaced by a 15-character identifier unique to the particular invocation of the do command. Finally, each instance of the pair && is replaced by an ampersand. Any other ampersand discovered in command_string causes an error message to be printed and the expansion to be terminated. Any argument supplied but not mentioned in a parameter designator is ignored.

2. control_argi is a character string argument to replace a parameter designator &i in command_string.

Usage as an Active Function

If the do command is called as an active function:

[do "command_string" arg1 ... argn]

then, instead of executing the resultant expanded command line, the do command passes it back as the value of the active function.

quote-stripping action to which the command processor subsequently subjects the expanded command line. If &qi is not located between quotes, or if control_argi contains no quotes, then the substitutions performed for &qi and for &i are identical.

If the parameter designator &ri is specified, then the substituted argument control_argi is placed between an additional level of quotes before having its quotes doubled. More precisely, if the parameter designator &ri is found nested to quote-level L, then 2**L quotes are inserted into the expanded line, control_argi is substituted into the expanded line with each of its quotes replaced by 2**(L+1)quotes, and then 2**L more quotes are placed following it. If argument control_argi is not supplied, then nothing is placed in the expanded line; this provides a way to distinguish between arguments that are not supplied and arguments that are supplied but null. If argument control_argi is present, then the expansions of &ri, and of &qi written between an additional level of quotes, are identical.

Accessing More than Nine Arguments

In addition to the normal parameter designators in which the argument to be substituted is specified by a single integer, do also allows the designators &(d...d), &r(d...d), and &q(d...d) where d...d denotes a string of decimal integers. An error message is printed and the expansion is terminated if any character other than 0 ... 9 is found between the parentheses.

Examples

The do command is particularly useful when used in conjunction with the abbreviation processor, the abbrev command. Consider the following abbreviations:

```
ADDPLI  do "fo &1.list;ioa_ ^|;pli &1;co"
AUTHOR  do "ioa_$nnl &1;status -author &1"
CREATE  do "cd &1;sis &1 re *.Demo rew Jay.*"
LIST    do "fo Jay.list;LISTAB;ws &1 LISTAC;co;dp -dl Jay.list"
LISTAB  do ".l"
LISTAC  "la;ls -dtm -a"
P       do "pl1 &1 -list &2 &3"
```

The command line:

```
ADDPLI alpha
```

expands to:

```
fo alpha.list;ioa_ ^|;pli alpha;co
```

do

while the command line:

```
P alpha -table
```

expands to:

```
pl1 alpha -list -table
```

This shows how references to unsupplied arguments get deleted. The abbreviation P2 is equivalent to P for 3 or fewer arguments. The command line:

P2 alpha -table -no_endpage -optimize

executes pl1 with the -optimize control argument, whereas:

P alpha -table -no_endpage -optimize

omits that last control argument.

Argument Substitution

Strings of the form &i in the exec_com segment are interpreted as dummy arguments and are replaced by the corresponding argument to the exec_com command. For instance, optional_arg1 is substituted for the string &1 and optional_arg10 is substituted for &10.

For argument substitution, by the string & followed by a number, by the string n,

The character & should be followed by a number, i, or by the string ec_name. If no corresponding optional_arg is provided, &i is replaced by the null string. The string &ec_name is replaced by the entryname portion of the exec_com pathname without the ec suffix. The string &0 is replaced by the pathname argument to exec_com, just as it was given to the command.

Argument substitution can take place in command lines, input lines or in control statements, since the replacement of arguments is done before the check for a control statement.

Control Statements

Control statements permit more variety and control in the execution of the command sequences. Currently the control statements are: &label, &goto, &attach, &detach, &input_line, &command_line, &ready, &print, &quit, &if, &then, and &else.

Control statements generally must start at the beginning of a line with no leading blanks. Exceptions to this rule are the &then and &else statements, that can appear elsewhere. Also when a control statement is part of a THEN_CLAUSE or an ELSE_CLAUSE, it does not have to start at the beginning of a line.

1. &label and &goto

These statements permit the transfer of control within an exec_com segment.

&label location	identifies the place to which a goto control statement transfers control. location is any string of 32 or fewer characters identifying the label.
&goto location	causes control to be transferred to the place in the exec_com segment specified by the label location. Execution then continues at the line immediately following the label.

The string &fi is ~~interpreted to mean~~ replaced by the ith through last arguments to exec_com. The string &n is replaced by the number of arguments supplied to exec_com.

Ver. 3
741022

Multics Change Request

MCR 1477
Page 1 of 1

TITLE: Change generate_mst to interpret object keyword correctly

AUTHOR: E. Stone

STATUS	DATE
Written	11/03/75
Status	A 11/11/75
Expires	05/11/76

DOCUMENTATION CHANGES

-Coded in: [X] PL/I [ ] ALM [ ] other- explain in DETAILED PROPOSAL
-Planned for System MR 3.1
-Fixes Bug Number(s)
-Documented in MTB
-User/Operations-visible Interface change? [ ] yes [X] no
-Incompatible change? [X] yes [ ] no
-Performance: [ ] Better [X] Same [ ] Worse
-Replaces MCR

Category (Check One)	
X	Lib. Maint. Tools
	Sys. Anal. Tools
	Sys. Prog. Tools
	355
	BOS
	Salvager
	Ring Zero
	Ring One
	SysDaemon/Admin.
	Runtime
	User Cmnd/Subr.

Document	Specify One or More
MPM (Vol, Sect.)	
PLMS (AN #)	51 Tools
MOSN (Sect.)	
MPAM (Sect.)	
MSAM (Sect.)	
Info Segs	
Other (Name)	
None (Reason)	

Objections/Comments:

Use these headings: SUMMARY, REASONS, IMPLICATIONS, DETAILED PROPOSAL (Optional)

SUMMARY:

Change the mst generator to write out the entire object segment when presented with the object keyword.

REASONS:

In order to implement pre-linking, segments which are used in all rings (such as pl1_operators_) must be represented in the hierarchy as standard object segments.