MULTICS TECHNICAL BULLETIN MTB-205

TO: Distribution

FROM: Joan Archer

DATE: 23 June 75

RE: Multics Change Requests

Enclosed are copies of Multics Change Requests which were approved from 1 June 75 to 15 June 75.

Ver. 4
750508

Multics Change Request

MCR 1177
Page 1 of

TITLE: Install new pl1_operators_ and trace

AUTHOR: R. A. Barnes

STATUS	DATE
Written	05/27/75
Status	A 06/03/75
Expires	12/03/75

-Coded in: ☒ PL/I ☒ ALM □ other- explain in DETAILED PROPOSAL
-Planned for System MR
-Fixes Bug Number(s)
-Documented in MTB 187
-User/Operations-visible Interface change? ☒ yes □ no
-Incompatible change? □ yes ☒ no
-Performance: □ Better □ 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
X	User Cmmd/Subr.

DOCUMENTATION CHANGES

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

Objections/Comments:

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

SUMMARY:

Install pl1_operators_ and trace with new interface.

REASONS:

trace presently uses 1 word/external entry of internal static in an object program to determine if the program is being traced. In order to reduce system overhead, the PL/I compiler soon will no longer be generating these internal static words, therefore trace will have to search its table to decide if a program is being traced.

New entry operators for separate_static have also been implemented. These will be invoked only by programs compiled with the -separate_static control argument.

IMPLICATIONS:

trace's performance will degrade since it will now have to search its table at every external call. System performance should improve, however, as the unnecessary internal static gets removed.

DETAILED PROPOSAL:

The separate_static entry operators will store a packed pointer to the static section at sp|28. The trace entry operators will always call trace$catch_pl1_ to decide if a program is being traced.

Installation will take place in three stages:

1. pl1_operators_ and trace will be installed in >sss. Al Berglund will be informed to exclude these 2 segments from any distribution tape made until step 3 is performed.

2. pl1_operators_ will be installed in the hardcore system.

3. pl1_operators_ will be deleted from >sss.

5/1/75

Name: trace

The trace command is a debugging tool that lets the user monitor all calls to a specified set of external procedures. The trace command modifies the standard Multics procedure call mechanism so that whenever control enters or leaves one of the procedures specified by the user, a debugging procedure is invoked. The user can request:

1. printing of the arguments at entry, exit or both.
2. stopping (by calling the command processor) at entry, exit or both.
3. changing the frequency with which tracing messages are printed (e.g. every 100 calls, after the 2000th call or only if the recursion depth is less than five).
4. executing a Multics command line at entry, exit or both.
5. metering of the time spent in the various procedures being monitored.

Usage: trace -control_arg-

where control_arg may be a pathname or a control argument.

pathname — add the procedure to the trace table with the tracing parameters from the Trace Control Template (TCT). If the procedure is already in table, the counters are reset and table, the current parameters in TCT are used.

For control arguments that affect the trace control template (TCT), the argument n, a number, must be present. The numbers in parentheses are the initial values.

-template — list the trace control template
-tp

-first n — begin monitoring on the nth call (1)
-ft n

-last n — stop monitoring after the nth call (9999999999)
-lt n

-every n — monitor every nth call (1)
-ev n

-before — call the command processor before calling the
-be n — traced procedure every n times. (0 = don't call)

-after n — call the command processor after calling the
-af n — the traced procedure. (0 = don't call)

-argument n — print the arguments every nth time the
-ag n — procedure is entered. (0 = don't print)

-depth n — maximum recursior depth to be monitored. (0, no
-dh n — limit)

-in — print the arguments only on entry. (yes)

-out	print the arguments only on exit. (no)
-inout	print the arguments on both entry and exit. (no)
-execute string -ex string	execute the Multics command line string when ever the procedure is monitored. ("" = no command)
-meter on	meter the time spent in the procedure. (off = no metering)
-meter off	stop metering the procedure.
-mt	(See Metering, below.)
-govern on	limit the recursion level for a procedure. (off = no governing)
-govern off	don't limit the recursion level for a procedure.
-gv	(Used after a -govern on request. See Recursion Limiting, below.)
-return_value on	print the return_value on exit. (off = no printing.)
-return_value off	don't print the return_value on exit.
-rv	(-rv yes, assumes entry is a function.)

For control arguments that affect procedures being traced, the argument e is an entry name or *. If an entry name is used, the control argument applies to that procedure. If * is used, the control argument is applied to all entries in the trace table.

-status e	print the trace parameters and counters for procedure e.
-status * -st *	print the procedures being monitored and their counters.
-reset e -rs e	set the number of calls and recursion depth of the specified procedure to zero.
-off e	stop monitoring the specified procedure. The procedure remains in the trace table and calls continue to be counted.
-on e	resume monitoring the specified procedure. Used after the -off control argument.
-remove e -rm e	remove the specified procedure from the trace table. Tracing can be removed at any time.

General control arguments:

-brief -bf	print a short form of the monitoring information.
-long -lg	print the long form of the monitoring information. (For use after the -brief control argument to restore the long form.)
-watch string -wh string	watch user specified locations and stop by calling the command processor when they change. (See the Watch Facility, below.)

Ver. 4
750508

Multics Change Request

MCR 1181
Page 1 of 14

TITLE: io_call command changes

AUTHOR: L. Johnson

STATUS	DATE
Written	05/19/75
Status	P 05/27/75 A 6/03/75
Expires	8/03/75

-Coded in: ☒ 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 ☐ no
-Incompatible change? ☒ yes ☐ no
-Performance: ☐ Better ☒ Same ☐ Worse
-Replaces MCR 1160

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.)	AG92
PLMS (AN #)	
MOSN (Sect.)	
MPAM (Sect.)	
MSAM (Sect.)	
Info Segs	
Other (Name)	
None (Reason)	

Objections/Comments:
Item 6 may go into the SWG

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

SUMMARY:

Implement a new io_call command with the following changes from the previous version:

1) support I/O operations to and from a segment (as in the old iocall command) as well as the terminal. This applies to get_chars, get_line, read_record, put_chars, write_record, rewrite_record operations.
2) additional abbreviations for operation names and open modes.
3) implementation of move_attach operation.
4) -brief control argument on modes operation.
5) change control operation to delete the -string argument and always use a null infoptr.
6) document the look_iocb, destroy_iocb, and print_iocb operations which already exist.

The attached MPM command on io_call reflects all the proposed changes.

Name: io_call, io

This command performs an operation on a designated I/O switch.

Usage

```
io_call opname switchname -args-
```

where:

1. opname designates the operation to be performed.

2. switchname is the name of the I/O switch.

3. args may be one or more arguments, depending on the particular operation to be performed.

The opnames permitted, followed by their alternate forms where applicable, are:

attach
close
control
delete_record, delete
detach_iocb, detach
destroy_iocb, destroy
find_iocb, find
get_chars
get_line
modes
move_attach, move
look_iocb, look
open
position
print_iocb, print
put_chars
read_key
read_length
read_record, read
rewrite_record, rewrite
seek_key, seek
write_record, write

Usage is explained below under a separate heading for each designated operation. The explanations are arranged functionally rather than alphabetically.

Unless otherwise specified, if a control block for the I/O switch does not already exist, an error message is printed on user_output and the operation is not performed. If the requested operation is not supported for the switch´s attachment/opening, an error message is printed on user_output.

The explanations of the operations cover only the main points of interest and, in general, treat only the cases where the I/O switch is attached to a file or device. For full details see the descriptions of the iox_ subroutine and the I/O modules in the MPM Subroutines and Section IV "Input and Output Facilities" of the MPM Reference Guide, respectively.

Operation: attach

The usage of attach is:

```
io_call attach switchname modulename -args-
```

where:

1. modulename is the name of the I/O module to be used in the attachment.

2. args may be one or more arguments, depending on what is permitted by the particular I/O module.

This command attaches the I/O switch using the designated I/O module. The attach description is the concatenation of modulename and args separated by blanks. The attach description must conform to the requirements of the I/O module.

If a control block for the I/O switch does not already exist, one is created.

Operation: detach

The usage of detach (or detach_iocb) is:

```
io_call detach switchname
```

This command detaches the I/O switch.

Operation: open

The usage of open is:

```
io_call open switchname mode
```

where mode is one of the following opening modes, which may be specified by its full name, or by an abbreviation:

stream_input, sti
stream_output, sto
stream_input_output, stio
sequential_input, sqi
sequential_output, sqo
sequential_input_output, sqio
sequential_update, squ
keyed_sequential_input, ksqi
keyed_sequential_output, ksqo
keyed_sequential_update, ksqu
direct_input, di
direct_output, do
direct_update, du

This command opens the I/O switch with the specified opening mode.

Operation: close

The usage of close is:

```
io_call close switchname
```

This command closes the I/O switch.

Operation: get_line

The usage of get_line is:

```
io_call get_line switchname -n- -control_args-
```

where:

1. n may be a decimal number greater than zero specifying the maximum number of characters to be read.

2. control_args may be selected from the following list:

   -segment path -offset-,
   -sm path -offset-
   specifies that the line read from the I/O switch is to be stored in the segment specified by pathname, at the location specified by offset.

   -nnl specifies that the newline character is to be deleted from the end of the line if it is present.

   -nl specifies that a newline character is to be added to the end of the line if one is not present.

   -lines specifies that the offset, if given, is measured in lines, rather than in characters. This control argument only has meaning if the -segment control argument is also specified.

This command reads the next line from the file or device to which the I/O switch is attached. If n is given, and the line is longer than n, then only the first n characters are read.

If the -segment control argument is not specified, the line read will be written onto the I/O switch user_output, with a newline character appended if one is not present and -nnl has not been specified.

If the -segment control argument is specified, the line will be stored in the segment named. If this segment does not exist, it will be created. If offset is specifed, the line will be stored at that position relative to the start of the segment. This is normally measured in characters, unless -lines has been used. If offset is omitted, the line is appended to the end of the segment. The bit count of the segment is always updated to a point beyond the newly added data.

Operation: get_chars

The usage of get_chars is:

```
io_call get_chars switchname n -control_args-
```

where:

1. n — is a decimal number greater than zero specifying the number of characters to read.

2. control_args — may be selected from the same list as described under the get_line operation.

This command reads the next n characters from the file or aevice to which the I/O switch is attached. The disposition of the characters read is the same as described under the get_line operation; that is, they are written upon user_output if the -segment control argument is not specified, or stored in a segment if the -segment control argument is specified.

Operation: put_chars

The usage of put_chars is:

```
io_call put_chars switchname -string- -control_args-
```

where:

1. string may be any character string.

2. control_args may be selected from the following list:

-segment path -length-,
-segment path -offset- -length-,
-sm path -length-,
-sm path -offset- -length-
specifies that the data for the output operation is to be found in the segment specified by pathname. The location and length of the data may be optionally described with offset and length parameters.

-nnl specifies that a newline character is not to be appended to the end of the output string.

-nl specifies that a newline character is to be added to the end of the output line if one is not present.

-lines specifies that offsets and lengths are measured in lines instead of characters.

The string parameter and the -segment control argument are mutually exclusive. If a string is specifed, the contents of the string will be the data output to the I/O switch. If the -segment control argument is specified, the data will be taken from the segment specified by path, at the offset and length given. If offset is omitted, the beginning of the segment is assumed. If length is omitted, the entire segment will be output.

If the I/O switch is attached to a device, this command transmits the characters from either the string of the segment to the device. If the I/O switch is attached to an unstructured file, the data is added to the end of the file. The -nl control argument is the default on a put_chars operation: a newline character will be added unless one is already present, or the -nnl control argument is specified.

Operation: read_record

The usage of read_record (or read) is:

```
io_call read_record switchname n -control_args-
```

where:

1. n is a decimal integer greater than zero specifying the size of the buffer to use.

2. control_args may be selected from the same list as described under the get_line operation.

This command reads the next record from the file to which the I/O switch is attached into a buffer of length n. If the -segment control argument is not specified, the record (or the part of it that fits into the buffer) is printed on user_output. If the -segment control argument is specified, the record will be stored in a segment as explained under the get_chars operation.

Operation: write_record

The usage of write_record (or write) is:

```
io_call write_record switchname -string- -control_args-
```

where:

1. string is any character string.

2. control_args may be selected from the same list as described under the put_chars operation.

This command adds a record to the file to which the I/O switch is attached. If the string parameter is specified, the record is equal to the string. If the -segment control argument is specified, the record will be extracted from the segment as described under the put_chars operation. In either case, the -nnl control argument is the default: a newline character is added only if the -nl control argument is specified. If the file is a sequential file, the record is added at the end of the file. If the file is an indexed file, the record's key must have been defined by a preceding seek_key operation.

Operation: rewrite_record

The usage of rewrite_record (or rewrite) is:

```
io_call rewrite_record switchname -string- -control_args-
```

where:

1. string is any character string.

2. control_args may be selected from the same list as described under the put_chars operation.

This command replaces the current record in the file to which the I/O switch is attached. The new record is either the string parameter, or is taken from a segment, as described under the write_record operation. The current record must have been defined by a preceding read_record, seek_key, or position operation as follows:

read_record current record is record read.

seek_key current record is record with the designated key.

position current record is the record preceding the record to which the file was positioned.

Operation: delete_record

The usage of delete_record (or delete) is:

```
io_call delete_record switchname
```

This command deletes the current record in the file to which the I/O switch is attached. The current record is determined as in rewrite_record above.

Operation: position

The usage of position is:

```
io_call position switchname type -n-
```

where:

1. type is -1, 0, or 1.

2. n is a decimal integer. It must be present if type is 0.

This command positions the file to which the I/O switch is attached. If type is -1, the file is positioned to its beginning, so that the next record is the first record (structured files), or so that the next byte is the first byte (unstructured files). If type is +1, the file is positioned to its end; the next record (or next byte) is at the end of file position. If type is 0, the file is positioned forwards ($n \geq 0$) or backwards ($n < 0$) over records (structured files) or lines (unstructured files). The number of records or lines skipped is determined by the absolute value of n.

In the case of unstructured files, the next byte position after the operation is at a byte immediately following a newline character (or at the first byte in the file or at the end of the file); and the number of newline characters moved over is the absolute value of n.

If the I/O switch is attached to a device, only forward skips (type=0, n ≥ 0) are allowed. The effect is to discard the next n lines input from the device.

Operation: seek_key

The usage of seek_key is:

```
io_call seek_key switchname key
```

where key is a string of ASCII characters with 0 ≤ length ≤ 256.

This command positions the indexed file to which the I/O switch is attached to the record with the given key. The record's length is printed on user_output. Trailing blanks in the key are ignored.

If the file does not contain a record with the specified key, it becomes the key for insertion. A following write_record operation adds a record with this key.

Operation: read_key

The usage of read_key is:

```
io_call read_key switchname
```

This command prints, on user_output, the key and record length of the next record in the indexed file to which the I/O switch is attached. The file's position is not changed.

Operation: read_length

The usage of read_length is:

```
io_call read_length switchname
```

This command prints, on user_output, the length of the next record in the structured file to which the I/O switch is attached. The file´s position is not changed.

Operation: control

The usage of control is:

```
io_call control switchname order
```

where order is one of the orders accepted by the I/O module used in the attachment of the I/O switch.

This command applies only when the I/O switch is attached via an I/O module that supports the control I/O operation. The command calls iox_$control (iocb_ptr, order, infoptr, code), where iocb_ptr designates the I/O switch. For full details see the description of iox_$control in the MPM Subroutines. The infoptr argument passed to iox_$control is always a null pointer.

Operation: modes

The usage of modes is:

```
io_call modes switchname -string- -control_arg-
```

1. string may be a sequence of modes separated by commas. The string must not contain blanks.

2. control_arg may be -brief or -bf

This command applies only when the I/O switch is attached via an I/O module that supports modes. The command prints, on user_output, the existing modes and, if string is given, sets the new modes as specified by string. Printing of the old modes is supressed if the -brief control argument is used.

If the switch name is user_i/o, the command refers to the modes controlling the user's terminal. See the I/O module tty_ subroutine description in the MPM Subroutines for an explanation of applicable modes.

Operation: find_iocb

The usage of find_iocb (or find) is:

```
io_call find_iocb switchname
```

This command prints, on user_output, the location of the control block for the I/O switch. If it does not already exist, the control block is created.

Operation: look_iocb, look

The usage of look_iocb (or look) is:

```
io_call look_iocb switchname
```

This command prints, on user_output, the location of the control block for the I/O switch. If the I/O switch does not exist, an error is printed.

Operation: move_attach

The usage of move_attach (or move) is:

```
io_call move_attach switchname switchname2
```

where switchname2 is the name of a second I/O switch.

This command moves the attachment of the first I/O switch (switchname) to the second I/O switch (switchname2). The origional I/O switch is left in a detached state.

Operation: destroy_iocb

The usage of destroy_iocb (or destroy) is:

```
io_call destroy_iocb switchname
```

This command destroys the I/O switch by deleting its control block. The switch must be in a detached state before this command is used. Any pointers to the I/O switch beco~~r~~me invalid.

Operation: print_iocb

The usage of print_iocb (or print) is:

```
io_call print_iocb switchname
```

This command prints, on user_output, all of the data in the control block for the I/O switch, including all pointers and entry variables.

er. 3
41022

Multics Change Request

MCR 1192
Page 1 of 1

TITLE: Fix bugs in FSDCT Initialization and Growth

AUTHOR: A. Kobziar

STATUS	DATE
Written	05/22/75
Status	A 06/03/75
Expires	12/03/75

-Coded in: ☒ PL/I ☐ ALM ☐ other- explain in DETAILED PROPOSAL
-Planned for System MR 2.2
-Fixes Bug Number(s) unreported
-Documented in MTB
-User/Operations-visible Interface change? ☐ yes ☒ no
-Incompatible change? ☐ yes ☒ no
-Performance: ☒ Better ☐ Same ☐ Worse
-Replaces MCR

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

DOCUMENTATION CHANGES

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

Objections/Comments:

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

SUMMARY:

Change the maximum number of saved FSDCT pages in a cold boot from 4 to the max FSDCT size in the header, 16. Change the salvager to supply any disk addresses missing due to a shutdown failure of a bootload which built or increased the FSDCT.

IMPLICATIONS:

A Dual Disk Subsystem configuration which interleaves both devices can now be handled, rather than crashing on a "out of E191" error.

DETAILED PROPOSAL:

Change the constant "fsdct_fms" in initialize_dims from 4 to 16.

Change init_salv_fsdct to look at the stored fsdct map size rather than the aste current length when deciding whether to grow the FSDCT, since the salvager's reused address checking may have already referenced the unassigned pages. A new FSDCT file map will be built in a temporary segment and then paged in again. This indirection is necessary because the FSDCT's file map is located in the first page of the FSDCT.

Ver. 4
750508

Multics Change Request

MCR 1193 .
Page 1 of 2 .

TITLE: Add move request to EDM

AUTHOR: J. Gintell

STATUS	DATE
Written	05/29/75
Status	A 06/03/75
Expires	12/03/75

-Coded in: ☒ 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 ☐ no
-Incompatible change? ☐ yes ☒ no
-Performance: ☐ Better ☒ 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.)	AG92 III
PLMS (AN #)	
MOSN (Sect.)	
MPAM (Sect.)	
MSAM (Sect.)	
Info Segs	
Other (Name)	
None (Reason)	

Objections/Comments:

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

SUMMARY:

Add a move request to EDM that can move text from one location in the segment to another. To keep the user interface simple, this operation is specified in terms of line number only.

REASONS:

It is extremely awkward to move text with EDM. This extension would be useful to all users of EDM. It would also permit EDM to be offered as part of FAST to replace most of the functions handled by the DTSS EDIT command.

DETAILED PROPOSAL:

The MPM description is as follows:

Move (move)

Format: move n m

Purpose: move m lines starting at line number n to after the current line. If the lines to be moved overlap the current lines in the segment to move, an error message is printed by EDM and the request is not performed.

Pointer: Set to immediately after the text moved.

Default: If *m* is not specified, 1 line is moved.

Example:

```
Before:      a
             b
             c
            →d
             e
             f

Request:     move 2 2

After:       a
             d
             b
            →c
             e
             f
```

Ver. 4
750508

Multics Change Request

MCR 1194
Page 1 of 1

TITLE: Add cobol_operators_ to system

AUTHOR: S. Webber

STATUS	DATE
Written	05/22/75
Status	A 06/03/75
Expires	12/03/75

-Coded in: □ 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 ☒ no
-Incompatible change? □ yes ☒ no
-Performance: □ Better ☒ 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.
X	Runtime
	User Cmmd/Subr.

DOCUMENTATION CHANGES

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

Objections/Comments:

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

SUMMARY:

Add a link to cobol_operators_$cobol_operators_ to the array of such in operator_pointers_.

REASONS:

The cobol compiler will soon generate code using cobol_operators_.

Ver. 4
750508

Multics Change Request

MCR 1195
Page 1 of 2

TITLE: Change dump_segment to call dump_seg_
AUTHOR: S. Webber

STATUS	DATE
Written	05/22/75
Status	[illegible] A 06/10/7
Expires	12/10/75

-Coded in: □ 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 □ no
-Incompatible change? ☒ yes □ no
-Performance: ☒ Better □ 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.)	X
PLMS (AN #)	
MOSN (Sect.)	
MPAM (Sect.)	
MSAM (Sect.)	
Info Segs	X
Other (Name)	
None (Reason)	

Objections/Comments:

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

SUMMARY:

1. Add a new subroutine, dump_seg_, which writes out data in the format that dump_segment does. Change dump_segment to call this subroutine.

2. Allow dump_segment to dump ring zero segments if the user has access to.

3. Change dump_segment to work if called as an active function.

REASONS:

1. dump_seg_ is a useful interface which other programs could call.

2. ring_zero_dump does not format the data well.

3. It is convenient to have an active function which returns the octal (or ascii) representation of a word.

06/19/75
Name: dump_seg_

The dump_seg_ subroutine can be used to format and write out specified regions of segments. The output format is the same as the dump_segment command.

Usage

```
declare dump_seg_ entry (ptr, ptr, fixed bin, fixed bin, bit (4));

call dump_seg_ (iocbp, datap, offset, count, format);
```

1) iocbp — is a pointer to the IOCB for the switch over which the data is to be printed.(Input)

2) datap — is a pointer to the first word of data to be dumped. (Input)

3) offset — is the offset value used only in printing the lefthand margin of numbers. (Input)

4) count — is the number of words to print out. (Input)

4) format — is the desired output mode. The mode is specified by turning on bits in the format argument as follows:

bit 1	short mode
bit 2	bcd mode
bit 3	ascii mode
bit 4	brief mode

If both the ascii and bcd bits are on, a default mode of brief only is used.
(Input)

(END)

Ver. 4
750508

Multics Change Request

MCR 1196
Page 1 of 2

TITLE: Update bound_segment_info_

AUTHOR: M. Weaver

STATUS	DATE
Written	05/27/75
Status	A 06/03/75
Expires	12/03/75

-Coded in: ☒ PL/I ☐ AIM ☐ other- explain in DETAILED PROPOSAL
-Planned for System MR
-Fixes Bug Number(s)
-Documented in MTB 187
-User/Operations-visible Interface change? ☐ yes ☒ no
-Incompatible change? ☐ yes ☒ no
-Performance: ☐ Better ☒ 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
X	User Cmmd/Subr.

DOCUMENTATION CHANGES

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

Objections/Comments:

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

SUMMARY:

Update several components of bound_segment_info_ to use the new object_info_ structure. Do other cleaning up specified below.

REASONS:

Many of these procedures need to use the static pointer from the new object_info_ structure. get_bound_seg_info_ and decode_definition_$full are given pointers to object_info_ structures and so must be changed to work with both versions. get_bound_seg_info_ and display_component_name must be converted to version 2 PL/I. display_component_name has a bug in code added to search sections other than text; this code should be deleted since the documentation only mentions searching the text anyway. object_info_ should be deleted because there is a duplicate in hardcore. old_print_link_info should be deleted because it is no longer needed (we don't have to worry about segments bound by the old binder any more). decode_definition_ must recognize the new definition class 4. interpret_link_ uses any length when copying names, needs to recognize the new *static section code for self-referencing links, and needs a new entry point for use with archived object segments (see attached documentation). print_bind_map should call iox_. print_link_info should no longer call old_print_link_info.

IMPLICATIONS:

Most of these procedures need to be installed soon after object_info_, before any translators begin generating the new object segment format. A case could be made for installing get_bound_seg_info_ and decode_definition_ before object_info_ since they are given a pointer to an object_info_ structure, but these entries are not widely used, especially outside the system, and I think could safely be updated soon after object_info_.

DETAILED PROPOSAL:

This MCR covers all components of bound_segment_info_ except for print_linkage_usage, which will be the subject of a future MCR.

The output of print_link_info and print_bind_map will indicate whether an object segment has separate static.

Entry: interpret_link_$tptr

This entrypoint differs from interpret_link_ only in that a pointer to the text section of the object segment is supplied by the caller. This entrypoint is useful if the object segment image does not start at the beginning of a segment.

Usage:

```
          declare interpret_link_$tptr entry(ptr, ptr, ptr, fixed bin(35));

          call interpret_link_$tptr(symb_ptr, link_ptr, text_ptr, code);
```

1) symb_ptr   is a pointer to a structure in which the symbolic information will be returned as above. (Input)

2) link_ptr   is a pointer to the link to be decoded. (Input)

3) text_ptr   is a pointer to the text section of the object segment in which the link resides. (input)

4) code   is a returned standard error code as above. (Output)

Ver. 4
750508

Multics Change Request

MCR 1197
Page 1 of 1

TITLE: Fix privileged quota setting in append

AUTHOR: A. Kobziar

STATUS	DATE
Written	05/29/75
Status	A 06/03/75
Expires	12/03/75

-Coded in: ☒ PL/I ☐ ALM ☐ other-explain in DETAILED PROPOSAL
-Planned for System MR 2.2
-Fixes Bug Number(s)
-Documented in MTB
-User/Operations-visible Interface change? ☐ yes ☒ no
-Incompatible change? ☐ yes ☒ no
-Performance: ☐ Better ☒ Same ☐ Worse
-Replaces MCR

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

DOCUMENTATION CHANGES

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

Objections/Comments:

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

SUMMARY:

Change append to use the correct directory pointer when setting quota for a privileged ring 1 process (i.e. reloader).

REASONS:

Append currently sets the quota on the parent rather than the newly created directory.

Ver. 4
750508

Multics Change Request

MCR 1198
Page 1 of 1

TITLE: Fix bug in BOS FMT command

AUTHOR: Noel I. Morris

STATUS	DATE
Written	5/30/75
Status	A 06/03/75
Expires	12/03/75

-Coded in: □ PL/I ☒ ALM □ other- explain in DETAILED PROPOSAL
-Planned for System MR
-Fixes Bug Number(s) 049-42
-Documented in MTB
-User/Operations-visible Interface change? □ yes ☒ no
-Incompatible change? □ yes ☒ no
-Performance: □ Better ☒ Same □ Worse
-Replaces MCR

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

DOCUMENTATION CHANGES

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

Objections/Comments:

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

Summary:
The FMT command fails to operate properly if the disk subsystem being formatted is not the subsystem on which BOS itself is residing.

Proposal:
Change an ALS instruction to a QLS instruction to make program work properly.

Implications:
AFDSC will be able to format their E191's.

Ver. 4
750508

Multics Change Request

MCR 1199
Page 1 of 1

TITLE: Remove "null *.*.*" acl from hardcore headers

AUTHOR: B. May (Phx)

STATUS	DATE
Written	05/27/75
Status	A 06/03/75
Expires	12/03/75

-Coded in: □ 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 □ no
-Incompatible change? □ yes ☒ no
-Performance: □ Better ☒ Same ☒ Worse
-Replaces MCR ____

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

DOCUMENTATION CHANGES

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

Objections/Comments:

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

SUMMARY:

Currently, the "header" control files for the hardcore system tape generator contain several gate entries that set "null *.*.*" on the segment being processed. This is redundant and inefficient, in that each new access to the gate must check the extra null ACL entry.

Ver. 4
750508

Multics Change Request

MCR 1200
Page 1 of 1

TITLE: Fix bug in BOS CORE Command

AUTHOR: Noel I. Morris

STATUS	DATE
Written	3/30/75
Status	A 06/03/75
Expires	12/03/75

-Coded in: □ PL/I ☒ ALM □ other-
explain in DETAILED PROPOSAL
-Planned for System MR
-Fixes Bug Number(s) C80671
-Documented in MTB
-User/Operations-visible
Interface change? □ yes ☒ no
-Incompatible change? □ yes ☒ no
-Performance: □ Better ☒ Same
□ Worse
-Replaces MCR

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

DOCUMENTATION CHANGES

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

Objections/Comments:

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

Summary:
A coding bug caused the BOS CORE command to ignore a specified tape number as an argument to the command and to always attempt to use tape drive 1. The same bug also caused the CORE command not to wait after a CORE RESTOR was performed.

Proposal:
Change a TZE instruction to a TNZ instruction.

Ver. 4
750508

Multics Change Request

MCR 1201
Page 1 of 1

TITLE: Change mail's program_interrupt feature

AUTHOR: S. Herbst

STATUS	DATE
Written	06/03/75
Status	A 06/10/75
Expires	12/10/75

-Coded in: ☒ 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 □ no
-Incompatible change? ☒ yes □ no
-Performance: □ Better ☒ 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.)	
PLMS (AN #)	
MOSN (Sect.)	
MPAM (Sect.)	
MSAM (Sect.)	
Info Segs	
Other (Name)	
None (Reason)	Not Mentioned

Objections/Comments:

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

SUMMARY: Change the mail command to respond to "Delete? yes" after printing was interrupted by program_interrupt bv deleting all the messages in the mailbox rather than just the ones that were printed.

REASONS: Both mail commands respond to program_interrupt by skipping to the query "Delete? " The old mail command responded to "yes" by deleting all the messages, so that if a user already knew the contents of his mailbox he could emp ty it out. The new mail command currently responds to "yes" by deleting only the messages that were printed before the interruption. The division between printed and non-printed messages is blurred by output buffering so that there still may be one or more messages deleted that were not actually printed.

Ver. 4
750508

Multics Change Request

MCR 1202
Page I of I

TITLE: fix ol_dump to work with new db_print

AUTHOR: Susan Barr

STATUS	DATE
Written	06/03/75
Status	A 06/10/75
Expires	12/10/75

-Coded in: ☒ PL/I ☐ ALM ☐ other-explain in DETAILED PROPOSAL
-Planned for System MR 2.2
-Fixes Bug Number(s) ____
-Documented in MTB ____
-User/Operations-visible Interface change? ☐ yes ☒ no
-Incompatible change? ☐ yes ☒ no
-Performance: ☐ Better ☒ 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
X	User Cmmd/Subr.

DOCUMENTATION CHANGES

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

Objections/Comments:

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

REASONS:

db_print uses an external data base for the output iocb ptr for calls to iox_$put_chars. ol_dump does not set the initial value of the iocb ptr before calling db_print.

PROPOSAL:

Remove the data base from db_print and pass the iocb pointer as a parameter. This change requires modification of these programs:

db_print
db_parse
ol_dump

Ver. 4
750508

Multics Change Request

MCR 1203
Page 1 of 1

TITLE: Remove temporary test provisions for Access Isolation
AUTHOR: A. Kobziar

STATUS	DATE
Written	06/04/75
Status	A 06/10/75
Expires	12/10/75

-Coded in: □PL/I ☒ALM □other-
explain in DETAILED PROPOSAL
-Planned for System MR 2.2
-Fixes Bug Number(s) ____
-Documented in MTB ____
-User/Operations-visible
Interface change? ☒ yes □ no
-Incompatible change? □ yes ☒ no
-Performance: □ Better ☒ Same
□ Worse
-Replaces MCR ____

Category (Check One)

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

DOCUMENTATION CHANGES

Document	Specify One or More
MPM (Vol, Sect.)	
PLMS (AN #)	
MOSN (Sect.)	
MPAM (Sect.)	
MSAM (Sect.)	
Info Segs	
Other (Name)	
None (Reason)	not documented

Objections/Comments:

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

REASONS:

The temporary provisions for testing Access Isolation are no longer needed. set_pds, which allowed arbitrary settings of access_authorization should be removed. The privilege bits in the template pds which were left on for everyone are now superfluous as the answering service now sets them as specified.

SUMMARY:

Delete set_pds; set the privilege bits in the template pds to off. (This must be installed after AS 8.0)

Ver. 4
750508

Multics Change Request

MCR 1204
Page 1 of 1

TITLE: Inform BOS of APT offset

AUTHOR: R. Mullen

STATUS	DATE
Written	06/02/75
Status	A 06/10/75
Expires	12/10/75

-Coded in: □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 ☒ no
-Incompatible change? □ yes ☒ no
-Performance: □ Better ☒ Same □ Worse
-Replaces MCR

Category (Check One)

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

DOCUMENTATION CHANGES

Document	Specify One or More
MPM (Vol, Sect.)	
PLMS (AN #)	73
MOSN (Sect.)	
MPAM (Sect.)	
MSAM (Sect.)	
Info Segs	
Other (Name)	
None (Reason)	

Objections/Comments:

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

SUMMARY: An unused, currently zero, location in tc_data will be used to store the offset of the APT. BOS will use the stored value, if nonzero, otherwise BOS will use 400(octal) as it currently does.

REASONS: This change makes BOS independent of all but the most drastic changes to the structure of tc_data. In particular it will allow the priority scheduler to be installed without a simultaneous installation of BOS.

IMPLICATIONS: BOS will be dependent on the contents of location 253(octal) in tc_data to find the offset of the APT. BOS is already dependent on the contents of location 327(octal) for apt_entry_size.

DETAILED PROPOSAL: See "SUMMARY"

Ver. 4
750508

Multics Change Request

MCR 1205
Page 1 of 1

TITLE: Fix Salvager's not flushing core and paging device
AUTHOR: A. Kobziar

STATUS	DATE
Written	06/05/75
Status	A 06/10/75
Expires	12/10/75

-Coded in: □PL/I □ALM □other-explain in DETAILED PROPOSAL
-Planned for System MR MIT 360
-Fixes Bug Number(s)
-Documented in MTB
-User/Operations-visible Interface change? □ yes ☒ no
-Incompatible change? □ yes ☒ no
-Performance: □ Better ☒ Same □ Worse
-Replaces MCR

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

DOCUMENTATION CHANGES

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

Objections/Comments:

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

REASONS:

Salvager does not flush core at the end of a salvage, and therefore fails to write to disk some pages still in core or on the paging device.

SUMMARY:

Add a pc$flush call at the end of a salvage.

Ver. 4
750508

Multics Change Request

MCR 1206
Page 1 of 1

TITLE: Fix bug in card reader DIM

AUTHOR: Noel I. Morris

STATUS	DATE
Written	75.06.05
Status	A 06/10/75
Expires	12/10/75

-Coded in: ☒ 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 ☒ no
-Incompatible change? ☐ yes ☒ no
-Performance: ☐ Better ☒ Same ☐ Worse
-Replaces MCR

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

DOCUMENTATION CHANGES

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

Objections/Comments:

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

Reason:
The "resetread" entry point of the card reader DIM contains an error in coding. This entry point has never been used before, but it is now required by the new card reading software.

Proposal:
Fix the bug in crz_dim$resetread.

Ver. 4
750508

Multics Change Request

MCR 1207
Page 1 of 1

TITLE: Change BOS Toehold

AUTHOR: Noel I. Morris

STATUS	DATE
Written	75.06.05
Status	A 06/10/75
Expires	12/10/75

-Coded in: □ 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 ☒ no
-Incompatible change? □ yes ☒ no
-Performance: □ Better ☒ Same □ Worse
-Replaces MCR ____

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

DOCUMENTATION CHANGES

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

Objections/Comments:

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

Proposal:
Change the toehold mechanism in BOS. Currently, setup, the BOS main control program, is read into memory only when a machine image is to be saved by the toehold. If the toehold is reentered during BOS operation, it is assumed that setup is already in memory. The mechanism should be changed to always read in a fresh copy of setup, whether or not a machine image is saved.

Reasons:
During BOS operation, the main control program sometimes gets clobbered. It is currently impossible to obtain a fresh copy of setup without destroying the saved machine image in BOS. The proposed change would enable operations to restore a fresh copy of setup without destroying the saved machine image. This would, in fact, occur automatically. Hence, no operational procedures need be changed.

Ver. 4
750508

Multics Change Request

MCR 1208
Page 1 of 1

TITLE: Multics COBOL Compiler
AUTHOR: T. Gerassimenko (CEO-BOSTON/SEC)

STATUS	DATE
Written	27 May 75
Status	A 06/10/75
Expires	12/10/75

-Coded in: ☒ PL/I □ ALM □ other- explain in DETAILED PROPOSAL
-Planned for System MR 3.0
-Fixes Bug Number(s) ____
-Documented in MTB ____
-User/Operations-visible Interface change? ☒ yes □ no
-Incompatible change? □ yes ☒ no
-Performance: □ Better □ 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.)	X
PLMS (AN #)	(planned)
MOSN (Sect.)	
MPAM (Sect.)	
MSAM (Sect.)	
Info Segs	
Other (Name)	Multics COBOL User's Guide, Multics COBOL Programmer's Manual
~~None (Reason)~~	

Objections/Comments:

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

SUMMARY: This initial release 1.0 of the COBOL compiler supports a subset of the 1974 ANSI COBOL as specified in the Multics COBOL '74 Product Functional Specification for phase 1 implementation.

The compiler accepts a source segment x.cobol, (which is prepared by a Multics text editor or alternatively from cards which are read into a segment), and produces an object segment x to execute under Multics, and a list segment x.list.

A COBOL program can call normal Multics subroutines using the CALL statement, and access normal files with the restrictions that the parameters must be character strings and the COPY files must have the suffix .incl.cobol.

REASONS: This is the first installation of a COBOL compiler for the Multics system (MR3.0) to satisfy marketing requirements.

IMPLICATIONS: All the names of the modules of this compiler currently starting with mc_ will be changed to cobol_ for the subsequent release of this compiler.

DETAILED: For detailed information reference:
1) MPM
2) Multics COBOL User's Guide
3) Multics COBOL Programmer's Manual

Ver. 4
750508

Multics Change Request

MCR 1209
Page 1 of 1

TITLE: Multics COBOL Compiler Support User Commands

AUTHOR: T. Gerassimenko

STATUS	DATE
Written	72 May 75
Status	A 06/10/75
Expires	12/10/75

-Coded in: ☒ PL/I ☐ ALM ☐ other- explain in DETAILED PROPOSAL

-Planned for System MR 3.0

-Fixes Bug Number(s) ____________

-Documented in MTB ____________

-User/Operations-visible Interface change? ☒ yes ☐ no

-Incompatible change? ☐ yes ☒ no

-Performance: ☐ Better ☐ 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.)	X
PLMS (AN #)	
MOSN (Sect.)	
MPAM (Sect.)	
MSAM (Sect.)	
Info Segs	
Other (Name)	
None (Reason)	

Objections/Comments:

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

SUMMARY:

Provide run-time support for COBOL programs.

The commands are:

rc (run_cobol)
dcr (display_cobol_run_unit)
scr (stop_cobol_ run)
fcs (format_cobol_source)
ccp (cancel_cobol_ program)
print_cobol_error_

For detailed information reference MPM.

Name: cobol

The cobol command invokes the COBOL compiler to translate a segment containing the text of a COBOL source program into a Multics object segment. A listing segment can also be produced. These segments are placed in the user's working directory. This command cannot be called recursively. For information on COBOL, refer to the *Multics COBOL Users' Guide*, Order No. AS43 and the *Multics COBOL Reference Manual*, Order No. AS44.

Usage

```
     cobol path -control_args-
```

where:

1. path — is the pathname of a COBOL source segment that is to be translated by the COBOL compiler. If path does not have a suffix of cobol, then one is assumed. However, the suffix cobol must be the last component of the name of the source segment.

2. control_args — may be chosen from the following list:

   -source, -sc — produces a line-numbered, printable ASCII listing of the program.

   -symbols, -sb — produces a source program listing (like the -source control argument), followed by a cross-reference listing of all data names defined in the program.

   -map — produces a source program listing with symbols (like the -symbols control argument), followed by a map of the object code generated by this compilation. The -map control argument produces sufficient information to allow the user to debug most problems online.

   -list, -ls — produces a source program listing with symbols (like the -symbols control argument), followed by an assembly-like listing of the compiled object program. Use of the -list control argument significantly increases compilation time and should be avoided whenever possible by using the -map control argument.

   -brief, -bf — causes error messages written to the user_output I/O switch to contain only an error number and statement identification once the full message has been given on the first occurrence. In the normal, nonbrief mode, an explanatory message is always written.

-severityi,
-svi

causes error messages whose severity is less than i (where i is 1, 2, 3, or 4) to not be written to the user_output I/O switch although all errors are written into the listing. If this control argument is not given, a severity level of 2 is assumed. See the description of severity levels under "Error Diagnostics" below.

-check, -ck is used for syntactic and semantic checking of a COBOL program. No code is generated.

-table, -tb generates a full symbol table for use by symbolic debuggers; the symbol table is part of the symbol section of the object program and consists of two parts: a statement table that gives the correspondence between source line numbers and object locations and an identifier table that contains information about every identifier actually referenced by the source program. The table appears in the symbol section of the object segment produced for the compilation.

-format, -fmt accepts source segment in the format acceptable to the format_cobol_source command.

The following control arguments are available, but are probably not of interest to every user.

-debug, -db leaves the work files generated by the compiler intact after a compilation. This control argument is used for debugging the compiler. The command cobol$clean_up may be used to discard these files. Also, this causes severity 4 errors to not unwind and abort the compilation, but rather to invoke a new level of the command processor at the point of the error.

-time, -tm prints the time (in seconds) and the number of page faults taken by each phase of the compiler; prints the total time at the end of the compilation. This is directed to the user_output I/O switch.

Notes

The only result of invoking the cobol command without control arguments is to generate an object segment.

A normal compilation produces an object segment and leaves it in the user's working directory. If an entry with that name existed previously in the directory, its access control list (ACL) is saved and given to the new copy of the object segment. Otherwise, the user is given re access to the segment with ring brackets v,v,v where v is the validation level of the process that is active when the object segment is created.

If the user specifies the -source, -symbols, -map, or -list control arguments, the cobol command creates a listing segment named path.list. The ACL is set as described for the object segment except that the user is given rw access to it when newly created. Previous copies of the object segment and the listing segment are replaced by the new segments created by the compilation.

severity level of the error.

If a listing is produced, the error messages appear interspersed with the lines of the source program. No more than 300 messages are printed in the listing.

Listing

The listing created by the cobol command is a line-numbered image of the source segment with diagnostics interspersed. This is followed by a cross-reference table of all the names defined within the program. Following the cross-reference table is the object code map, which gives the starting location in the text segment of the instructions for each statement in the program. The map is sorted by ascending storage locations. Finally, the listing contains an assembly-like list of the object code produced. The executable instructions are grouped under an identifying header, which contains the source statement that produced the instruction. Opcode, pointer-register, and modifier mnemonics are printed alongside the octal instruction. If the address field of the instruction uses the IC (self-relative) modifier, the absolute text location corresponding to the relative address is printed on the remarks field of the line.

Name: cancel_cobol_program, ccp

The cancel_cobol_program command causes one or more programs in the current COBOL run unit to be cancelled. Cancelling consists of ensuring that the next time the program is invoked within the run unit, its data is in an initial state. Any files that have been opened by the program and are still open are closed and the COBOL data segment is truncated. Refer to the run_cobol command for information concerning the run unit and the COBOL runtime environment.

Usage

```
     cancel_cobol_program names -control_arg-
```

where:

1. names — are the reference names of COBOL programs that are active in the current run unit. If the name specified in the PROG-ID statement of this program is different from name_i argument, this argument may be in the for refname$prog-id.

2. control_arg — may be -retain_data or -retd to leave the data segment associated with the program intact for debugging purposes.

Notes

The results of the cancel_cobol_program command and the execution of the CANCEL statement from within a COBOL program are similar. The only difference is that if a name_i argument is not actually a component of the current run unit, an error message is issued and no action is taken; for the CANCEL statement, no warning is given in such a case.

To maintain the value of the program's data for debugging purposes, the -retd control argument should be used. The data associated with the cancelled program is in its last used state; it is not restored to its initial state until the next time the program is invoked in the run unit.

Refer to the following related commands.

display_cobol_run_unit, dcr
stop_cobol_run, scr
run_cobol, rc

Name: display_cobol_run_unit, dcr

The display_cobol_run_unit command displays the current state of a COBOL run unit, i.e., most basically, which programs comprise the run unit. Optionally, more detailed information may be displayed concerning active files, data location, and other aspects of the run unit. Refer to the run_cobol command for information concerning the run unit and the COBOL runtime environment.

Usage

```
display_cobol_run_unit -control_args-
```

where control_args may be chosen from the following list:

-long, -lg
: causes more detailed information about each COBOL program in the run unit to be displayed.

-files
: displays information about the current state of the file that have been referenced during the execution of t current run unit.

-all, -a
: includes information about programs that have been cancelled during the execution of the run unit.

Note

Refer to the following related commands:

run_cobol, rc
stop_cobol_run, scr
cancel_cobol_program, ccp

Name: format_cobol_source, fcs

The format_cobol_source command converts pseudo free-form COBOL source to the standard fixed-format COBOL source expected by the COBOL compiler.

Usage

```
format_cobol_source path1 path2
```

where:

1. path1 is the pathname of the input segment containing pseudo free-form COBOL source. If path does not have a suffix of cobol, one is assumed. However, the suffix cobol must be the last component of the name of the input segment.

2. path2 is the pathname of the output segment that contains the converted fixed-format COBOL source. The cobol suffix is optional for the path2 argument; however, the output segment will have the cobol suffix. If the specified path2 argument is not found, a segment is created and given the path2 argument plus the cobol suffix as its name. If path2 is the same segment as path1, the converted output does not replace the input and an error message is printed.

Note

Pseudo free-form COBOL source statements are translated as follows:

```
Pseudo free-form COBOL                  Output
!column1                      !cclumn7  !column12

*XXX                          *XXX
a*XXX                          XXX
d*XXX                         d         XXX
da*XXX                        dXXX
/XXX                          /XXX
-XXX                                   -XXX
YYYXXX                                  YYYXXX
```

where X is any character and YYY is any three characters except:

1. *XX
2. a*X
3. d*X
4. da*
5. /XX
6. -XX

Name: print_cobol_error_

The print_cobol_error_ subroutine allows the COBOL programmer to display the cause and location of a runtime error. It is meaningful only when called from within a USE procedure in the DECLARATIVE section of a COBOL program. The error information displayed pertains to the error causing the current execution of the USE procedure. This is identical to the messages that would have been printed on the terminal before aborting the program (i.e., signalling the "error" condition) had no USE procedure been provided.

If the main entry point is used, the error information is displayed through the user_output I/O switch.

Usage

```
     call "print_cobol_error_".
```

Entry: print_cobol_error_$switch

This entry point outputs the error information to a specified I/O switch.

Usage

```
     01 switch-name pic x(32).

     call "print_cobol_error_$switch" using switch-name.
```

where switch-name is the name of an I/O switch that is open for output. This includes user_output and error_output, as well as the I/O switch associated with any open external COBOL file, i.e., the internal-file-name as specified in the SELECT clause of the ENVIRONMENT DIVISION. (Input)

Name: run_cobol, rc

The run_cobol command explicitly initiates execution of a COBOL run unit in a specified "main program". This command is not needed to execute COBOL object programs on Multics; it is used to simulate an environment in which more traditional COBOL concepts may be easily defined. This command cannot be called recursively.

Usage

```
run_cobol name -control_args-
```

1. name — is the reference name or pathname of the "main program" in which execution is to be initiated. If a pathname is given, then the specified segment is initiated with a reference name identical to the entryname portion of the pathname. Otherwise, the search rules are used to locate the segment. If the name specified in the PROG-ID statement of the COBOL program (i.e., the entry point name) is different than the current reference name of the object segment, then the name specified here must be in the form A$B where A is the pathname or reference name of the segment and B is the PROG-ID as defined in the IDENTIFICATION DIVISION of the source program.

2. control_args — can be chosen from the following:

   -cobol_switch *n*, -cs *n* — sets one or more of the eight COBOL-defined "external switches" on, where *n* is a number from 1 to 8 (or a series of numbers separated by spaces) that corresponds to the numbered external switch. At the outset of the run unit, the default setting of these external switches is off. (The eight external switches are defined in *The Multics COBOL Reference Manual*, Order No. AS44.)

   -no_stop_run, -nsr — avoids establishment of a handler for the stop_run condition. (See "Notes" below.)

Notes

This command enables the user to explicitly define and start execution of a COBOL run unit. A run unit is either explicitly started via execution of the run_cobol command or implicitly started by the execution of a COBOL object program either by invocation from command level or from a call by another language program. A run unit is stopped either by the execution of the STOP RUN statement in a COBOL object program or by invocation of the stop_cobol_run command. For the duration of time after a run unit is started and before it is stopped, it is said to be active. All COBOL programs executed while a run unit is active are considered part of that run unit.

A run unit is a subset of a Multics process; it is stopped when the process is ended. Also, when all programs contained in a run unit are cancelled, the run unit is stopped (refer to the cancel_cobol_program command). Only one run unit may be active at any given time in a process; thus, the run_cobol command cannot be invoked recursively. Additionally, if a run unit has been started implicitly (as described above), the run_cobol command may not be used until that run unit has been stopped, i.e., the run_cobol command does not terminate a currently active run unit.

The explicit creation of a run unit via the run_cobol command provides the following:

1. The establishment of a "main program", i.e., a program from which control does not return to the caller. The EXIT PROGRAM statement when encountered in such a program has no effect, as required in the COBOL definition of that verb. An implicitly started run unit has no "main program". The EXIT PROGRAM statement in all programs contained in such a run unit always causes control to be returned to the caller even if the caller is a system program, e.g., the command processor

2. The presetting of the COBOL external-switches. These switches are initialized off unless otherwise specified by the -cobol_switch control argument.

3. The ability to control the action taken when a STOP RUN statement in a COBOL object program is executed. The action normally taken for STOP RUN is to cancel all programs in the run unit closing any files left open. After this has been done, the data associated with any of the programs is no longer available. Thus in a debugging environment, it may be useful to redefine the action taken for STOP RUN. When the run unit is explicitly initiated via the run_cobol command, the STOP RUN statement causes the signalling of the stop_run conditior for which a handler is established that performs the normal action described above. If the -nsr control argument is specified, the handler is not established, thus allowing the user to field the signal himself using other Multics commands. For example the execution of the STOP RUN statement:

```
on stop_run "scr -retd" run_cobol progname -nsr
```

   causes the scr (stop_cobol_run) command to be executed with the control argument to retain data. If the user has not provided a handler himself for stop_run and specifies the -no_stop_run control argument, an unclaimed signal results.

The name given to the run_cobol command in which execution is to begin does not necessarily have to be a COBOL object program. It may be of any language that can provide a meaningful interface with COBOL programs (e.g., PL/I, FORTRAN).

Refer to the following related commands:

display_cobol_run_unit, dcr
stop_cobol_run, scr
cancel_cobol_program, ccp

Name: stop_cobol_run, scr

The stop_cobol_run command causes the termination of the current COBOL run unit. Refer to the run_cobol command for information concerning the run unit and the COBOL runtime environment.

Usage

```
stop_cobol_run -control_arg-
```

where the control_arg may be -retain_data or -retd to leave the data segments associated with the programs comprising the run unit intact for debugging purposes.

Notes

The results of the stop_cobol_run command and the execution of the STOP RUN statement from within a COBOL program are identical. Stopping the run unit consists of cleaning up all files that have been opened during the execution of the current run unit, and ensuring that the next time a program that was a component of this run unit is invoked, its data is in an initial state.

To maintain the value of all data referenced in the run unit in its last used state, the -retd control argument should be used.

Refer to the related commands:

display_cobol_run_unit, dcr
cancel_cobol_program, ccp
run_cobol, rc

Error Diagnostics

The COBOL compiler can diagnose and issue messages for about 800 different errors. These messages are graded in severity as follows:

1 Observation. Compilation continues without ill effect.

2 Warning. The compiler attempts to remedy the situation and continues, possibly without ill effect. The assumptions the compiler makes in remedying the situation, however, do not necessarily guarantee the right results.

3 Fatal--an uncorrectable but recoverable error. That is, the program is definitely in error and no meaningful object code can be produced, but the compiler can continue executing and diagnosing further errors.

4 Unrecoverable error. The compiler cannot continue beyond this error. A message is printed and control is returned to the cobol command. The command writes an abort message into the error_output I/O switch and returns to its caller.

As indicated above, the user can set the severity level so that he is not bothered by minor error messages. He can also specify the -brief control argument so that the message is shorter. Since the default severity level is 2, the user must explicitly specify the -severity1 (or -sv1) control argument when he invokes the cobol command in order to have observation messages printed. Neither the -severity nor -brief control argument has any effect on the contents of the listing segment if one is produced.

An example of an error message in its long form is:

```
     22         use after error procedure on extend.
                                             1
     ** 1 5-250 A use procedure has already been associated with this processing
mode.
```

If the -brief control argument is specified and message 5-250 has previously been given in its long form, the user instead sees:

```
     22         use after error procedure on extend.
                                             1
     ** 1 5-250
```

In the second case, the user could look up error number 5-250 in Appendix A of the Multics COBOL Users' Guide and get the full message (or of course he could refer to the previously printed message). If the user had set his severity level to 3, he would have seen no message at all. Notice that the number of asterisks immediately preceding the error indicator corresponds to the